MASTER EN SCIENCES INFORMATIQUES

Contribution au développement d'un système de pilotage pour la conception d'un schéma conceptuel d'informations

Gattelier, Marie-Hélène ; Bertinchamps, Jean-Pierre

Award date:
1986

Awarding institution:
Universite de Namur
Faculte d'informatique

FACULTES UNIVERSITAIRES NOTRE DAME DE LA PAIX (NAMUR)

Institut d'Informatique

ANNEE ACADEMIQUE 1985-86

CONTRIBUTION AU DEVELOPPEMENT D'UN SYSTEME DE PILOTAGE POUR LA CONCEPTION D'UN SCHEMA CONCEPTUEL D'INFORMATIONS

Mémoire présenté pour l'obtention du grade de :

Licencié et Maître en Informatique

Promoteurs : F. Bodart
J-L. Hainaut

Présenté par : Marie-Hélène Gatellier
Jean-Pierre Bertinchamps

Nous désirerions avant tout, remercier les personnes qui ont contribués à la réalisation de ce mémoire :

Monsieur Bodart, notre promoteur qui nous a attentitvement suivis et conseillés tout au long de notre travail.

Monsieur Hainaut, notre co-promoteur, pour les conseils et les remarques qu'il nous a prodigués lors de la réalisation du mémoire

Monsieur Kouloumdjiam et son équipe pour l'excellent accueil et le bon encadrement de travail qu'ils nous ont réservés.

Nos parents et tous ceux qui nous ont supportés et soutenus pendant ces quelques mois de travail.

OBJECTIF ET CONTENU

Les problèmes inhérents à l'utilisation croissante de l'ordinateur pour la conception d'applications de systèmes d'information, nous ont fourni les motivations nécessaires pour contribuer au développement d'un outil dont le but est d'apporter une aide véritable dans ce domaine.

Etant donné l'ampleur qu'a pris le modèle entité - association durant ces dernières années où il est devenu partie intégrante de nombreuses méthodologies de conception de base de données, il n'est pas étonnant qu'un certain nombre d'outils logiciel d'aide à la modélisation, basés sur ce modèle, aient été développés [de.13, de.14, de.18 à de.24].

Ces divers systèmes possèdent quelques points communs. S'ils possèdent tous, un ensemble de primitives pour la définition et/ou la modification d'un modèle de données, et des outils de conversion de ce modèle vers une base de données, peu d'entre eux possèdent un véritable outil capable d'aider le concepteur du modèle à exprimer au mieux la sémantique de son modèle [de.13].

Ce mémoire a pour objectif d'apporter des éléments au développement d'un outil de ec type, outil qui devrait s'intégrer dans le cadre du logiciel IDA.

Au cours d'une première phase, nous avons analysé la littérature traitant de ce sujet, littérature présente en annexe dans une bibliographie classée par matière.

La seconde phase a consisté en la réalisation d'un stage dans un centre européen de recherhces (LYON) où nous avons abordé les problèmes de conception d'un système expert. On pourra trouver en annexe l'étude d'un interface-utilisateur et de l'analyse de la cohérence de contraintes d'intégrité rélisée au cours de ce stage.

De retour en Belgique, nous avons abordé la réalisation du mémoire à proprement parlé où nous avons dans un premier temps défini précisément les propriétés souhaitables d'un modèle entité-association, avant d'entamer une réflexion sur des outils qui permettent d'aboutir à cette forme.

Remarque :

Une telle approche constitue une vision particulière du pilotage qui se déduit des propriétés du modèle. Une telle démarche ne prend pas en compte la démarche d'un analyste ce qui peut être considéré comme une limitation au travail réalisé.

Nous avons choisi de vous présenter en première partie de ce travail la forme souhaitée d'un schéma entité-association et sa représentation dans l'environnement IDA. Dans une seconde partie, nous nous sommes attachés à décrire les fondements et les options de base d'un outil d'aide. On trouvera en annexe les développements techniques relatifs à ces deux parties.

De plus nous supposerons que le lecteur est un familier du modéle entité-association et des modèles associés tel que le modèle relationnel.

PLAN DU TRAVAIL

PREMIERE PARTIE :

LE MODELE ENTITE - ASSOCIATION
ET
SA REPRESENTATION EN DSL

CHAPITRE 1 Forme générale d'un schéma entité - association

CHAPITRE 2 Forme canonique d'un schéma entité - association

CHAPITRE 3 Représentation en DSL d'un schéma entité - association

SECONDE PARTIE :

SYSTEME DE PILOTAGE POUR LA CONCEPTION D'UN SCHEMA CONCEPTUEL D'INFORMATIONS

CHAPITRE 4 Les différentes formes de pilotage et leurs problèmes

CHAPITRE 5 Contrôle a posteriori : règles de contrôle et scénario

CHAPITRE 6 Introduction à l'étude du pilotage interactif

TROISIEME PARTIE :

CONCLUSION

BIBIOGRAPHIE

ANNEXES

PREMIERE PARTIE :

LE MODELE ENTITE-ASSOCIATION
ET
SA REPRESENTATION EN DSL

INRODUCTION

Cette première partie de notre travail consistera à définir la forme canonique désirée pour le modèle entité-association (1) . Cette forme canonique est constituée d'un certain nombre de propriétés que devra respecter le schéma entité-association en sortie du système de pilotage. Cette forme canonique définit le type de modèle entité-association étudié dans ce travail.

Cette étape est très importante car elle servira de base pour l'étape suivante. En effet, une fois la définition de la forme finale désirée pour un schéma e-a précisée, nous pourrons déterminer la série de tests qui permettront de constater si le schéma e-a en notre possession respecte bien les règles specifiées lors de cette première étape, c'est-à-dire que nous pourrons déterminer les tests de validation, bases du système de pilotage.

En vue d'atteindre cet objectif, nous allons, dans une première phase, effectuer une étude du modèle entité - association général, c'est-à-dire l'ensemble des possibilités offertes à l'utilisateur pour la modélisation de son système d'informations. Lors d'une seconde phase, l'ensemble des propriétés que devrait respecter un schéma entité - association pour être sous forme canonique sera défini. Il s'agit donc d'un ensemble de restrictions qui sont apportées par rapport aux possibilités de la forme générale, et ce, dans le but d'améliorer l'expression de la sémantique de ce schéma.
Les propriétés de cette forme canonique ont été choisies et classées en fonction de 3 critères souvent présents dans la littérature, c'est-à-dire la complétude, le contrôle de redondance, et la cohérence.

Ce n'est que dans une troisième phase, que nous replacerons dans le cadre de ce travail, c'est-à-dire l'atelier-logiciel IDA, la représentation du schéma entité - association dans le langage DSL.

---

(1) Nous désignerons par "modèle entité-association" (e-a) ce que l'on définit généralement dans la littérature par l'expression "modèle entité-association-attribut" (e-r-a).

PLAN

PREMIERE PARTIE : LE MODELE ENTITE - ASSOCIATION ET SA REPRESENTATION EN DSL

CHAPITRE 1 FORME GENERALE D'UN SCHEMA ENTITE-ASSOCIATION

CHAPITRE 2 FORME CANONIQUE D'UN SCHEMA ENTITE - ASSOCIATION

CHAPITRE 3 REPRESENTATION D'UN SCHEMA ENTITE - ASSOCIATION EN DSL

CHAPITRE 1 FORME GENERALE D'UN SCHEMA ENTITE-ASSOCIATION

1.0 Introduction

Dans ce chapitre, nous allons spécifier les différents concepts du modèle entité - association qui permettent de décrire un système d'informations. De plus, nous décrirons l'ensemble des éléments qui peuvent être pris en compte pour la spécification de ces différents concepts.

1.1 Spécification possible pour les différents concepts du modèle e-a

1.1.1 Type d'entité

Définition :

Une entité est une chose concrète ou abstraite appartenant au réel perçu à propos de laquelle on veut enregistrer des informations.

La spécification d'un type d'entité peut, en général, comporter les éléments suivants :

a) Le nom du type d'entité

b) Synonyme(s) : tous les noms possibles pour le type d'entité.

c) Description : définition constitutive du type d'entité

d) Durée de vie : condition d'existence d'une entité dans le système d'informations.

e) 0 à N Attributs : caractéristiques du type d'entité

f) 0 à M Identifiants : mécanismes qui permettent d'identifier univoquement une occurrence d'un type d'entité.

Cet identifiant peut être constitué de :

- 0, 1 ou plusieurs attributs de ce type d'entité.

- 0, 1 ou plusieurs noms de rôle (1) liant d'autres types d'entité au type identifié via un type d'association.

g) Cardinalité : nombre maximum d'occurrences du type d'entité pendant une période donnée.

h) D'autres contraintes d'intégrité qui portent sur les types d'entité seront spécifiées au point 1.2.

ILLUSTRATION :

Type d'entité	:	CLIENT
Synonymes	:	CLI, CLT
Description	:	Toute personne physique ou morale dont au moins une commande passée a permis de l'identifier.
Durée de vie	:	5 ans
Attributs	:	numéro-client, nom-client, adresse
Identifiants	:	. numéro-client . nom-client + adresse
Cardinalité	:	5000

REMARQUE :

La spécification d'un type d'entité peut être complétée, à titre documentaire, par des informations telles que :

- le responsable du type d'entité, c'est-à-dire la personne responsable de la définition et des opérations autorisées sur ce type d'entité.

- l'aspect statistique de la cardinalité :

  Il serait judicieux d'indiquer la distribution du nombre d'occurrences en plus du nombre maximun d'occurrences d'un type entité, d'autant plus que la distribution peut inclure le concept de nombre maximun.

(1) cf. point 1.1.3

1.1.2 Attribut

Définition :

Caractéristique ou qualité d'un type d'entité ou d'un type d'association.

La spécification d'un attribut peut, en général, comporter les éléments suivants :

a) Le nom de l'attribut

b) Le type d'entité ou le type d'association qu'il qualifie.

c) Synonyme(s) : tous les noms possibles pour cet attribut

d) Description : définition de l'attribut.

e) Durée de vie : période durant laquelle l'attribut peut prendre une valeur significative.

f) Format de l'attribut : description de la représentation de l'attribut.

g) Domaine de valeurs : ensemble des valeurs que peut prendre cet attribut.

h) Unité de mesure

i) Simple ou répétitif : un attribut est simple si pour une occurrence d'un type d'entité ou d'association, il ne peut prendre qu'une seule valeur. Dans le cas contraire, il est répétitif.

j) Décomposable ou élémentaire :

un attribut est décomposable si à une occurrence d'un type d'entité ou d'association, il fait correspondre un groupe de valeurs de types différents et peut être décomposé, en autant d'attributs qu'il y a de types différents dans le groupe de valeurs. Dans ce cas, il conviendra de spécifier le nom des attributs qui caractérisent ce dernier.
Dans le cas contraire, il est élémentaire.

j) Obligatoire ou facultatif :

un attribut est obligatoire s'il est contenu au moins une fois au sein de l'élément qu'il décrit. Dans le cas contraire, il est facultatif.

k) Dérivable ou non dérivable :

un attribut est dérivable s'il peut être obtenu à partir d'un ou plusieurs autres attributs.

l) D'autres contraintes d'intégrité qui portent sur les attributs seront spécifiées au point 1.2

ILLUSTRATION :

Attribut	:	ADRESSE
Synonyme	:	ADR
Description	:	Adresse d'un CLIENT, lieu de livraison des produits commandés.
Durée de vie	:	1 an
Format	:	100 caractères
Domaine de valeurs	:	Alphanumérique
Simple		
Décomposable en	:	Code-postal, localité, rue, numéro
Obligatoire		
Non dérivable		

REMARQUE :

Il serait intéressant de compléter cette spécification par la probabilité qu'un attribut prenne une valeur nulle (valeur inconnue).

La spécification d'un attribut peut être complétée, à titre documentaire, d'informations telles que :

. Source de valeur de l'attribut dans le système d'informations.

ex: ADRESSE vient du BON-DE-COMMANDE (message)

1.1.3 Type d'association

Définition :

Une association est définie par une correspondance entre deux ou plusieurs entités (non nécessairement distinctes) où chacune assume un rôle donné.
L'existence d'une association est contingente à l'existence des entités qu'elle met en correspondance.

La spécification d'un type d'association peut, en général, comporter les éléments suivants :

a) Le nom du type d'association

b) Synonyme(s) : tous les autres noms possibles pour ce type d'association

c) Description : définition constitutive du type d'association

d) Durée de vie : conditions d'existence d'une association dans le système d'information. La période d'existence déterminée par ces conditions doit être contenue dans celles déterminées pour les entités sur lesquelles porte l'association.

e) Degré du type d'association : nombre de types d'entité reliés par ce type d'association.

f) 2..N Rôle(s) : le ou les rôles joués par chaque type d'entité qui participe à ce type d'association.

Chaque rôle est caractérisé par :

f.1) Un nom

f.2) Une connectivité :

nombre minimum d'occurrences de ce type d'association auquel doit participer toute occurrence d'un type d'entité et nombre maximum d'occurrences possibles du type d'association auquel peut participer toute occurrence d'un type d'entité.

f.3) Le type d'entité qui joue ce rôle

g) Cardinalité : nombre maximun d'occurrences de ce type d'association à une période donnée.
Cette cardinalité est déductible à partir de la cardinalité des types d'entité reliés au type d'association et de la connectivité associée à ces types d'entité, excepté dans le cas où cette connectivité est indéterminée.

h) Attributs : caractéristiques de ce type d'association

i) Identifiant : groupe formé par :

- 1 ou plusieurs noms de rôle des types d'entité qui participent à ce type d'association.

- 0, 1 ou plusieurs attributs de ce type d'association.

J) D'autres contraintes d'intégrité qui portent sur les types d'association seront spécifiées au point 1.2.

ILLUSTRATION :

Type d'association	:	PASSATION
Synonymes	:	PASS
Description	:	Une association "PASSATION" représente un engagement contractuel d'un CLIENT au moyen d'une COMMANDE
Durée de vie	:	2 mois
Degré	:	2
Rôles	:	. passe, 1-N, CLIENT . est passée par, 1-1, COMMANDE
Cardinalité	:	9000
Attribut	:	date-passation
Identifiant	:	. passe + est passé par

REMARQUE :

La spécification d'un type d'association peut être complétée, à titre documentaire, par des informations telles que :

. le responsable du type d'association, c'est-à-dire de sa définition et des opérations autorisées sur ce type.

. un aspect statistique de la cardinalité et de la connectivité :

Il serait judicieux de compléter la spécification de la connectivité et de la cardinalité par la distribution du nombre d'occurrences, d'autant plus que le concept de nombre maximum d'occurrences peut être inclus dans la distribution. Le point de vue existence pouvant être exprimé comme la probabilité qu'une occurrence d'un type d'entité participe à une occurrence de ce type d'association.

1.2 Contraintes d'intégrité

1.2.0 Introduction

L'utilisation du modèle entité - association permet certes de définir une structure d'informations, mais comme tout modèle il possède une capacité limitée de représentation du réel. C'est-à-dire qu'il ne permet pas d'exprimer la totalité des propriétés sémantiques dont le concepteur du système d'informations voudrait qu'il soit tenu compte afin de garantir la cohérence et la validité de la base de données.

Ces différentes propriétés qui complètent la description obtenue en termes de types d'entité, de types d'association, et d'attributs ont pour effet de limiter les occurrences possibles des structures d'informations.

Compte-tenu des objectifs poursuivis dans ce travail, nous nous proposons d'étudier les contraintes d'intégrité. Dans un premier temps, ces contraintes ont été classées selon leur type et dans un second temps, selon l'objet sur lequel elles portent.

REMARQUE :

La classification proposée ne se veut pas exhaustive, elle reprend uniquement les classes de contraintes d'intégrité les plus usuelles.

Chacune de ces classes sera illustrée par un exemple de contrainte d'intégrité basée sur un schéma entité - association que vous trouverez en annexe C.

1.2.1 Classes de contraintes d'intégrité de type statique

Au cours de ce paragraphe, seules les contraintes d'intégrité de type statique, c'est-à-dire les propriétés qui doivent être vérifiées à tout moment par les différentes occurrences des concepts du schéma entité - association, sont reprises.

1.2.1.1 Contraintes portant sur les types d'entité

a) Contrainte d'existence : condition qui lie la validité de l'existence d'une entité à d'autres éléments de la structure de données

ex : Une commande ne peut exister que si elle porte sur au moins un produit présent dans le catalogue.

b) Contrainte totale pour les sous-types :

toute occurrence du type d'entité doit être une occurrence d'au moins un de ses sous-types.

ex : Tout produit doit être de type électro-ménager ou hifi ou bois/verre ou peinture ou électricité.

Définition :

Nous dirons qu'un type d'entité Y est un sous-type d'un type d'entité X si et seulement si chaque occurrence du type d'entité Y est également une occurrence du type d'entité X.

Le processus de spécification fondamental qui est utilisé pour définir des sous-types d'entité est l'abstraction, et parmi les nombreux aspects de ce dernier, citons plus particulièrement :

la généralisation

ou

la spécialisation

La généralisation consiste à extraire d'un certain nombre de types d'entité la description d'un type plus général qui reprend les propriétés communes mais qui ignore certaines différences dans la description de ces types d'entité.

La spécialisation est un processus ayant l'effet opposé, c'est-à-dire que cela consiste à créer de nouveaux types d'entité où seront introduits des détails supplémentaires par rapport à la description du type d'entité existant.

L'idée de base de cette méthodologie est qu'un modèle peut-être construit en modélisant, en premier lieu, en termes de types d'entité, les concepts les plus généraux, et ensuite, en traitant des sous-cas au travers de types d'entité plus spécialisés.

Ce processus de spécification peut amener le concepteur à définir une hiérarchie de sous-types.
Dans cette hiérarchie, deux cas de figures peuvent se présenter :

. Cette hiérarchie est une arborescence stricte: c'est-à-dire qu'un type d'entité ne peut être au plus que sous-type d'un seul autre type d'entité.

  Cette optique a pour inconvénient que dans un certain nombre de cas, il sera nécessaire de doubler un certain nombre de types d'entité afin de conserver les propriétés d'arbre du graphe des sous-types, c'est-à-dire créer de le redondance.

. Cette hiérarchie n'est pas une arborescence : c'est-à-dire qu'un type d'entité peut être sous-type d'un nombre quelconque de types d'entité. Dans ce cas, il est nécessaire de spécifier des critères de sous-typage.

De plus, il convient de signaler un autre avantage que peut apporter le sous-typage.
En effet, cette relation de sous-type possède comme caractéristique la propriété d'héritage :

- Lorsqu'il s'agit de l'héritage des propriétés du type d'entité "père" du graphe, uniquement, nous parlerons d'héritage faible.

- Par contre, si les sous-types d'entité héritent également des types d'association au sein desquels le type d'entité "père" joue un rôle, nous parlerons d'héritage fort.

De plus, cette propriété d'héritage permet d'éviter la reproduction des caractéristiques communes à tous les sous-types d'entité au sein de chacun d'entre-eux.

c) Contrainte d'exclusion de sous-types :

contrainte spécifiant qu'une occurrence du type d'entité, reprise dans un ou plusieurs sous-types donnés, ne peut plus être reprise dans un ou plusieurs autres sous-types que ceux donnés

ex : Si un produit est de type électro-ménager, il ne peut être de type bois/verre, peinture, électricité.

d) Contrainte d'inclusion de sous-types :

contrainte spécifiant qu'une occurrence d'un type d'entité, reprise dans un ou plusieurs sous-types donnés doit être reprise dans un ou plusieurs autres sous-types.

ex : Si un produit est de type hifi, il est également de type électro-ménager.

REMARQUE :

Les deux contraintes précédentes ne sont que des cas particuliers des contraintes d'exclusion et d'inclusion pour les associations (cf. point 1.2.1.3 b,c)

1.2.1.2 Contrainte portant sur les attributs

a) Contrainte de valeurs : définit soit l'ensemble des valeurs que peut prendre un attribut, soit l'ensemble des valeurs que peut prendre cet attribut en fonction des valeurs prises par d'autres attributs (dérivabilité).

ex : Le montant d'une commande doit toujours être supérieur à 0.

b) Dépendance fonctionnelle : étant donné un type d'entité ou d'association,un groupe d'attributs B dépend fonctionnellement d'un groupe d'attributs A, si à tout moment, à chaque valeur de A correspond au plus une valeur de B

ex : NUMERO-CLIENT détermine fonctionnellement ADRESSE.

c) Contrainte d'existence conditionnelle :

détermine la présence obligatoire d'un attribut en fonction de la réalisation de conditions.

ex : Si q-stock = 0 alors date-app doit être présente.

1.2.1.3 Contraintes portant sur les types d'association

a) Contrainte d'existence : contrainte qui lie la validité d'existence d'une association à d'autres éléments de la structure de données.

ex : Une occurrence de ATTENTE ne peut exister que si elle porte sur une commande dont au moins un produit a une q-stock = 0.

b) Contrainte d'exclusion de rôle :

la participation d'une occurrence d'un type d'entité dans une occurrence d'un type d'association exclut sa participation dans une occurrence d'un autre type d'association

ex : Les occurrences de COMMANDE qui participent à une occurrence de ATTENTE doivent être différentes des occurrences de COMMANDE qui participent à une occurrence de FACT-COM.

c) Contrainte d'inclusion de rôle :

la participation d'une occurrence d'un type d'entité dans une occurrence d'un type d'association inclut sa participation dans une occurrence d'un autre type d'association

ex : L'ensemble des produits qui jouent le rôle "est de type H" est inclus dans l'ensemble des produits qui jouent le rôle "est de type EM".

d) Contrainte d'égalité : l'ensemble des occurences d'un type d'entité jouant un rôle donné dans un type d'association, doit être égal à l'ensemble des occurrences d'un même type d'entité mais jouant un rôle différent dans un autre type d'association

ex : L'ensemble des commandes passées par les clients est égale à l'ensemble des commandes constitué par l'union des commandes en attente et des commandes facturées.

e) Dépendance fonctionnelle : étant donné un type d'association un rôle ou un groupe de rôles B dépend fonctionnellement d'un rôle ou d'un groupe de rôles A, si à tout moment, à chaque occurrence de A correspond au plus une occurrence de B.

ex : passé par détermine fonctionnellement passe.

1.2.2 Classe de contraintes d'intégrité de type dynamique

Dans ce chapitre, seules les contraintes de type dynamique, c'est-à-dire les propriétés qui définissent la validité des changements d'état de la base de données, seront prises en compte à l'exception des propriétés définies au niveau statique.

1.2.2.1 Contraintes portant sur les attributs

a) Contrainte d'existence conditionnelle :

détermine les attributs obligatoires en fonction de la réalisation de conditions données.

ex : L'attribut BON-CLIENT est obligatoire si la moyenne annuelle des montants commandés croît et dépasse 60000.

REMARQUE:

Les deux types de contraintes qui suivent sont valables dans deux cas différents :

- pour une valeur d'un attribut
- pour plusieurs valeurs d'un même attribut

Dans les deux types de contrainte suivant le premier exemple se rapporte au premier cas et le suivant au second cas.

b) Contrainte de domaine faisant intervenir un seul attribut :

ex : Une contrainte pourrait être :

1) date-app ne peut changer que de la manière suivante :

valeur inconnue -> valeur date
ou valeur date -> valeur inconnue

2) la moyenne annuelle des commandes doit être supérieure de plus de 10% à la moyenne annuelle des commandes de l'année dernière.

c) Contrainte de domaine faisant intervenir plusieurs attributs d'un même type d'entité ou d'association :

ex : Une contrainte pourrait être :

1) La q-stock des différents produits ne peut diminuer que par multiple de 10 unités.

2) La somme des quantités commandées d'un produit de type bois/verre ne peut être supérieure à la somme des quantités commandées de ce même produit pour l'année précédente.

d) Contrainte de domaine faisant intervenir plusieurs attributs de types d'entité ou de types d'association différents :

ex :

Une contrainte pourrait être :

1) Le taux de remise consenti sur une facture pour un client donné ne peut croître ou décroître que dans la même proportion que la moyenne des montants commandés par ce client.

1.3 Contrainte générale associée au schéma entité-association

- Règle d'unicité des noms

Dans la forme brute du schéma entité-association, il existe cependant une contrainte à respecter :

Unicité des noms de type d'entité, de type d'association, de rôle parmi l'ensemble des concepts d'un schéma entité - association

- Justification

En général, un schéma conceptuel d'informations est construit sur base de ressources différentes telles que des documents écrits, des interviews, ressources issues principalement de l'étude d'opportunité.

L'étude de ces ressources selon le procédé de construction progressive d'un schéma permet d'éviter, en grande partie, l'extrême confusion qui résulte souvent de la recherche systématique dans un ensemble de documents de tous les types d'entités et de leurs attributs, ensuite de tous les types d'association, etc...

Cependant, ce procédé qui consiste à mettre en évidence, pour chaque proposition additionnelle prise en compte, les éléments nouveaux qu'elle peut contenir, tels que les types d'entité, d'association, ..., n'assure pas totalement l'absence d'une certaine confusion. En effet, cette méthode n'empêche en rien, par exemple, de voir apparaître plusieurs objets (1) différents portant le même nom, au sein du modèle des informations.

Ceci peut aussi bien être le signe de la présence de deux objets différents qui n'en forment qu'un, en réalité, que le signe d'une future confusion à éviter par la qualification des noms.

C'est pourquoi nous demandons l'unicité des noms d'objets, dès la forme générale, afin d'éviter l'apparition d'homonynie(s).

Par contre, nous ne demandons pas l'unicité des noms d'attribut parmi l'ensemble des attributs d'un schéma e-a. En toute rigueur, par définition même d'un attribut qui correspond à une relation fonctionnelle définie sur un ensemble d'entités ou d'associations et un ensemble de valeurs, on ne peut que requérir l'unicité des noms d'attribut.

---

(1) Par objet, nous entendons soit un type d'entité, soit un type d'association ou un rôle

---

Mais pratiquement, dans un très grand nombre de cas, on n'établit pas de distinction stricte entre l'attribut et le domaine de valeurs. C'est pourquoi dans la forme générale ,on peut ne pas admettre la règle d'unicité des noms d'attribut.

1.4 Schéma entité-association du modèle e-a

1.4.0 Notation

Pour faciliter la compréhension du lecteur, nous utiliserons les lettres majuscules pour représenter les types d'entité, les types d'association et les attributs.
Par contre, il sera fait usage de lettres minuscules pour la représentation d'entités ,d'associations et de valeurs d'attribut.

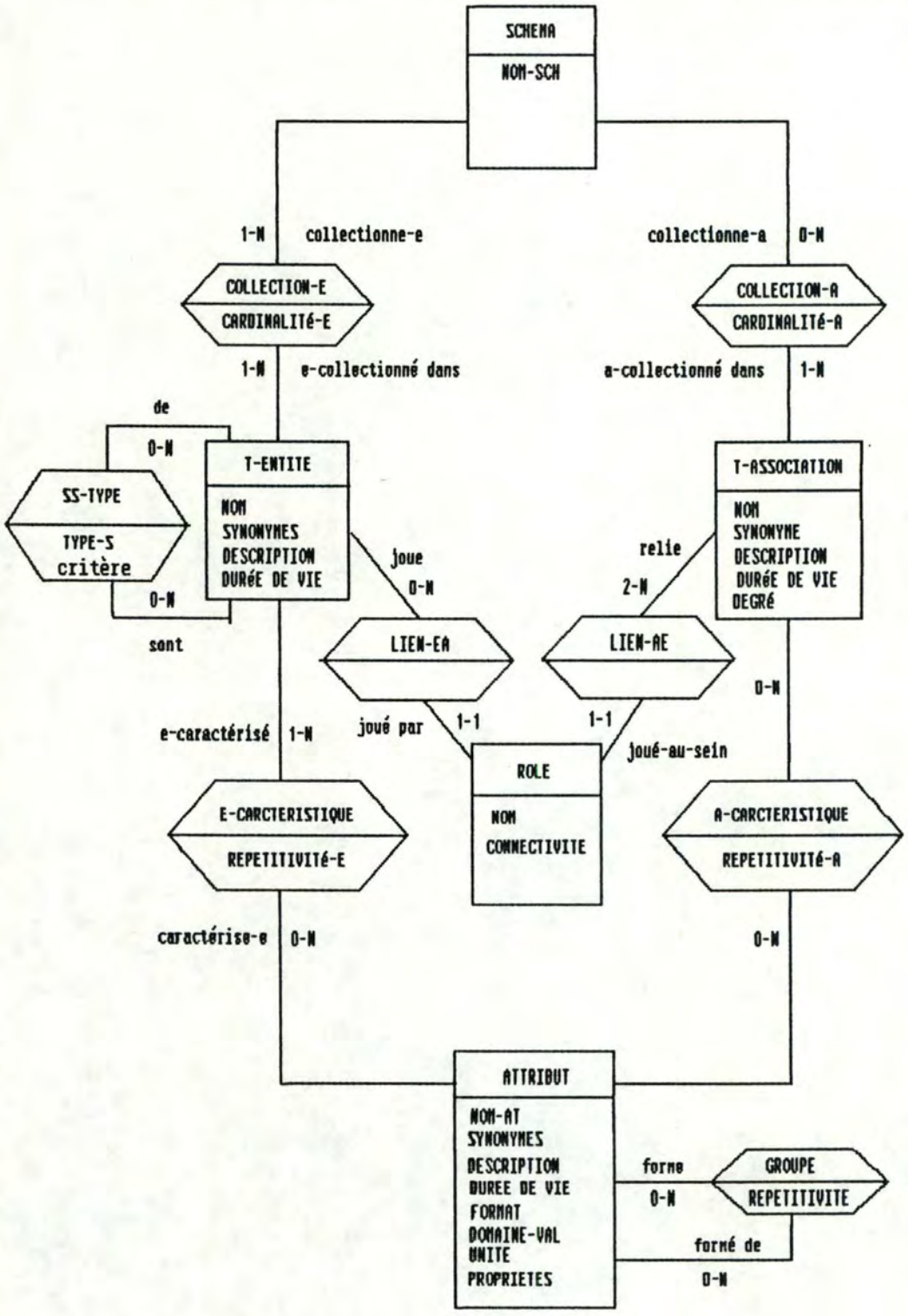
D'autre part, la convention adoptée pour la représentation graphique des concepts du modèle e-a n'est autre que celle introduite par CHEN P.P. [er.01] et communément employée dans la littérature.

1.4.1 Le modèle entité-association

1.4.1.1 Introduction

Au cours de ce point, nous allons représenter graphiquement le méta-schéma entité - association. En d'autres mots, nous allons faire le schéma entité - association du modèle entité - association que nous avons décrit au cours de ce chapitre.

Pour ce faire, nous allons considérer le modèle e-a comme un système d'informations à modéliser et nous allons essayer de dégager les types d'entité, les types d'association qui les relient et leurs attributs respectifs, sur base des points précedents de ce chapitre. Afin de simplifier le travail, nous ne tiendrons pas compte des contraintes d'intégrité.



Remarque : - Le domaine de valeur de "propriété" est défini en extension par :

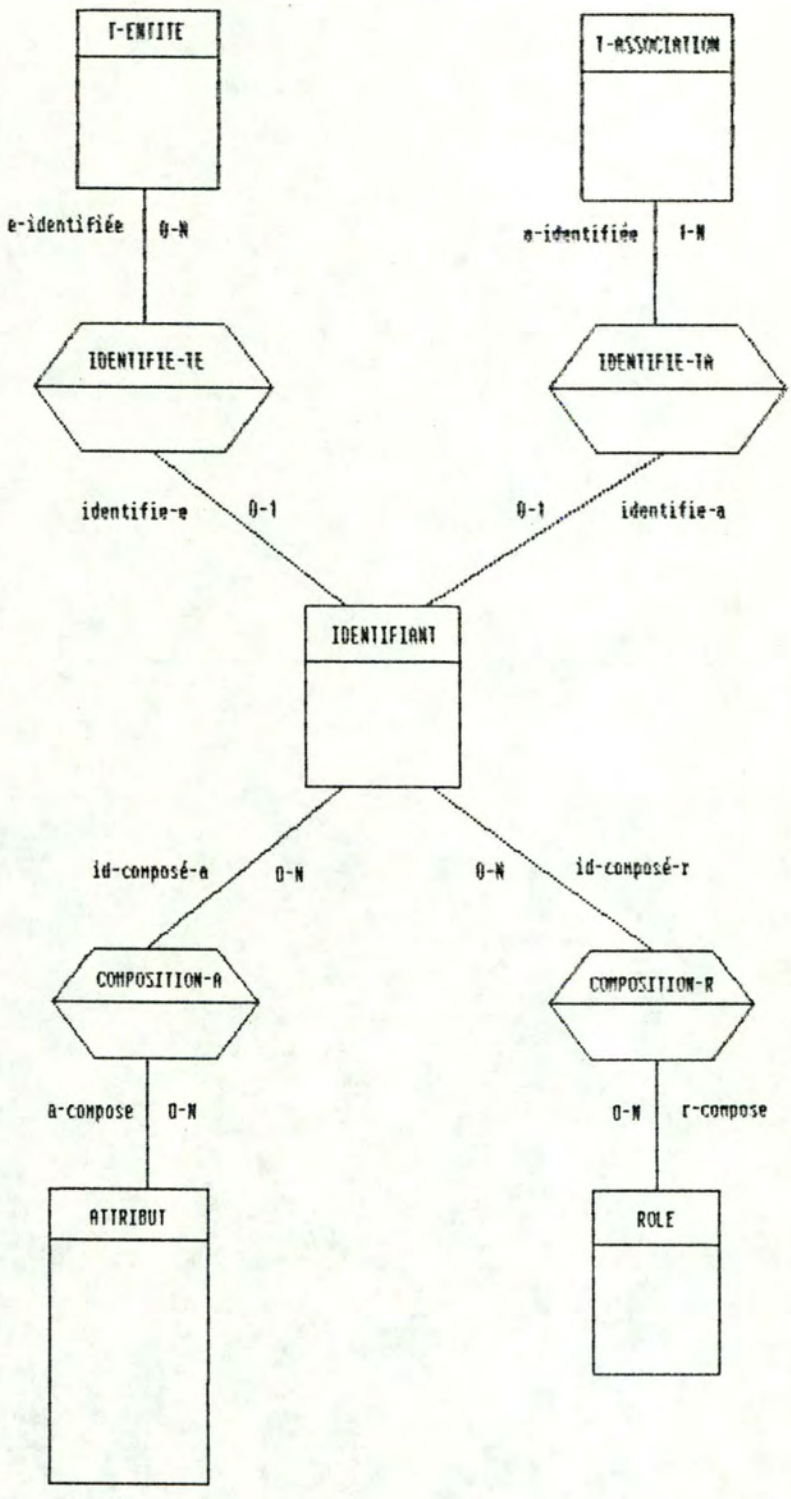
simple ou répétitif, élémentaire ou décomposable, obligatoire ou facultatif, dérivable ou non

Contrainte : - Le domaine de valeur de "type-s" est défini en extension par :

faible ou fort

- Un attribut doit au moins caractériser un type d'entité, un type d'association ou un attribut décomposable

1.4.2 Le modèle d'identification

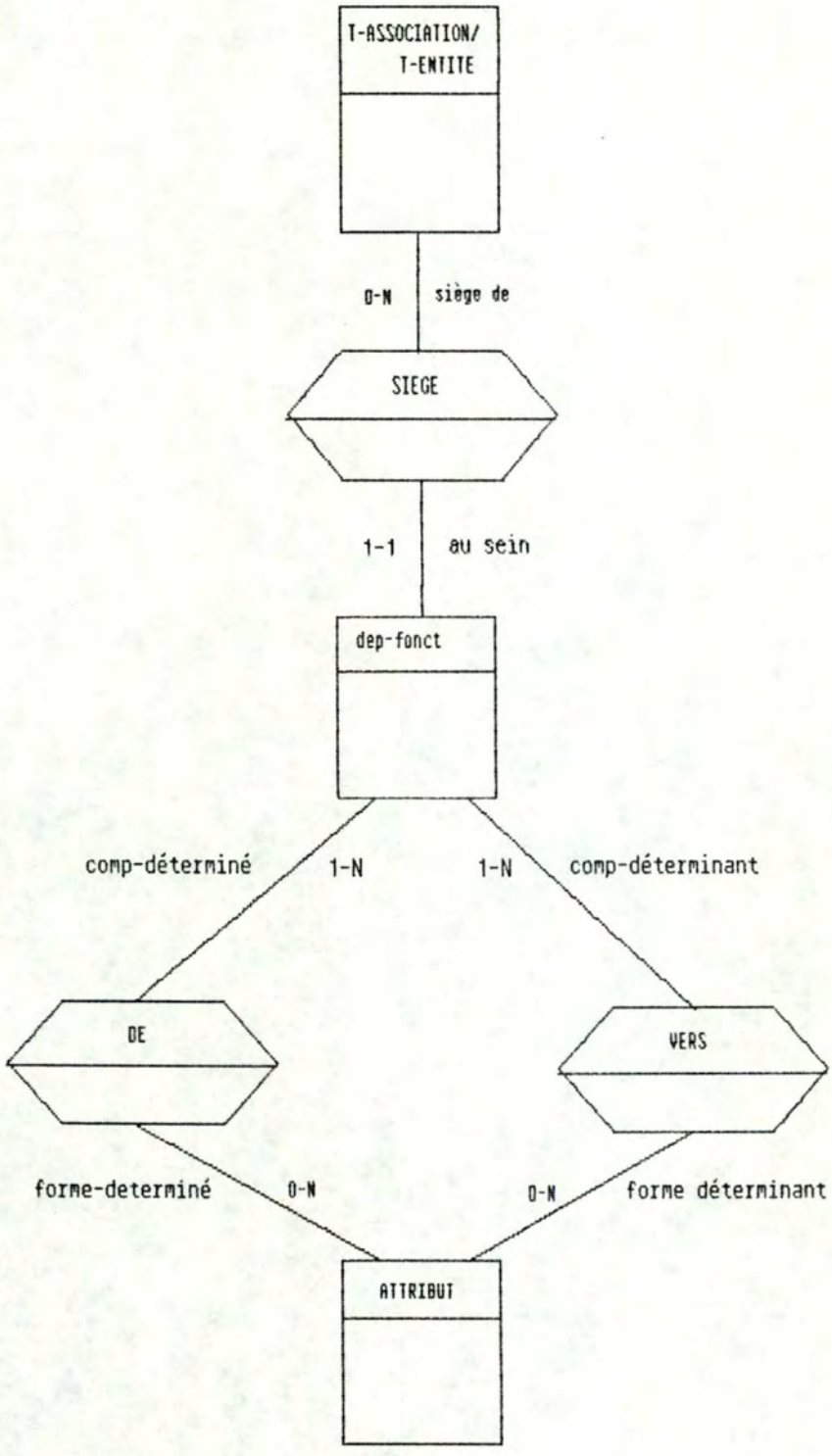


Contraintes : - un identifiant d'un type d'entité doit être composé d'au moins un attribut ou d'un rôle

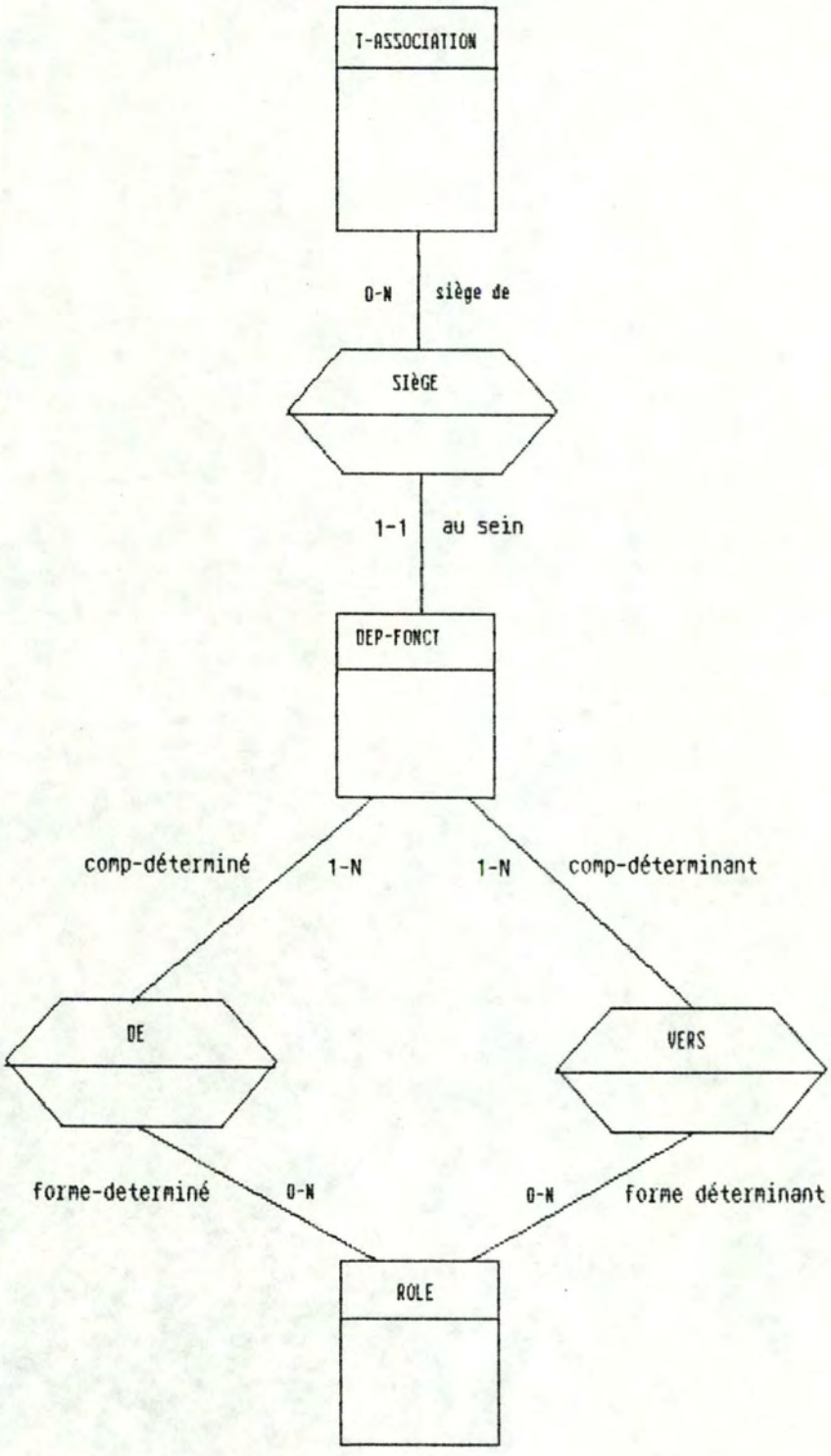
- un identifiant d'un type d'association doit être composé d'au moins un rôle

- un identifiant doit au moins identifier un type d'entité ou un type d'association

1.4.3 Le modèle des dépendances fonctionnelles



Contrainte : - Un attribut ne peut former à la fois un déterminé et un déterminant au sein d'une même dépendance fonctionnelle



Contrainte : - Un rôle ne peut former à la fois un déterminé et un déterminant au sein d'une même dépendnace fonctionnelle

CHAPITRE 2 FORME CANONIQUE DU SCHEMA ENTITE - ASSOCIATION

Préambule

Avant d'aborder réellement l'étude de la forme canonique d'un schéma entité - association, nous allons définir un certain nombre de concepts qui interviendront tout au long de ce chapitre. Il s'agira des différentes formes normales et du modèle relationnel auquel se rapportent celles-ci.

Le modèle relationnel

[rm.04, rm.05, rm.06, rm.07, rm.08]

Le modèle relationnel est basé sur la notion de table qui constitue une manière simple de voir les données.

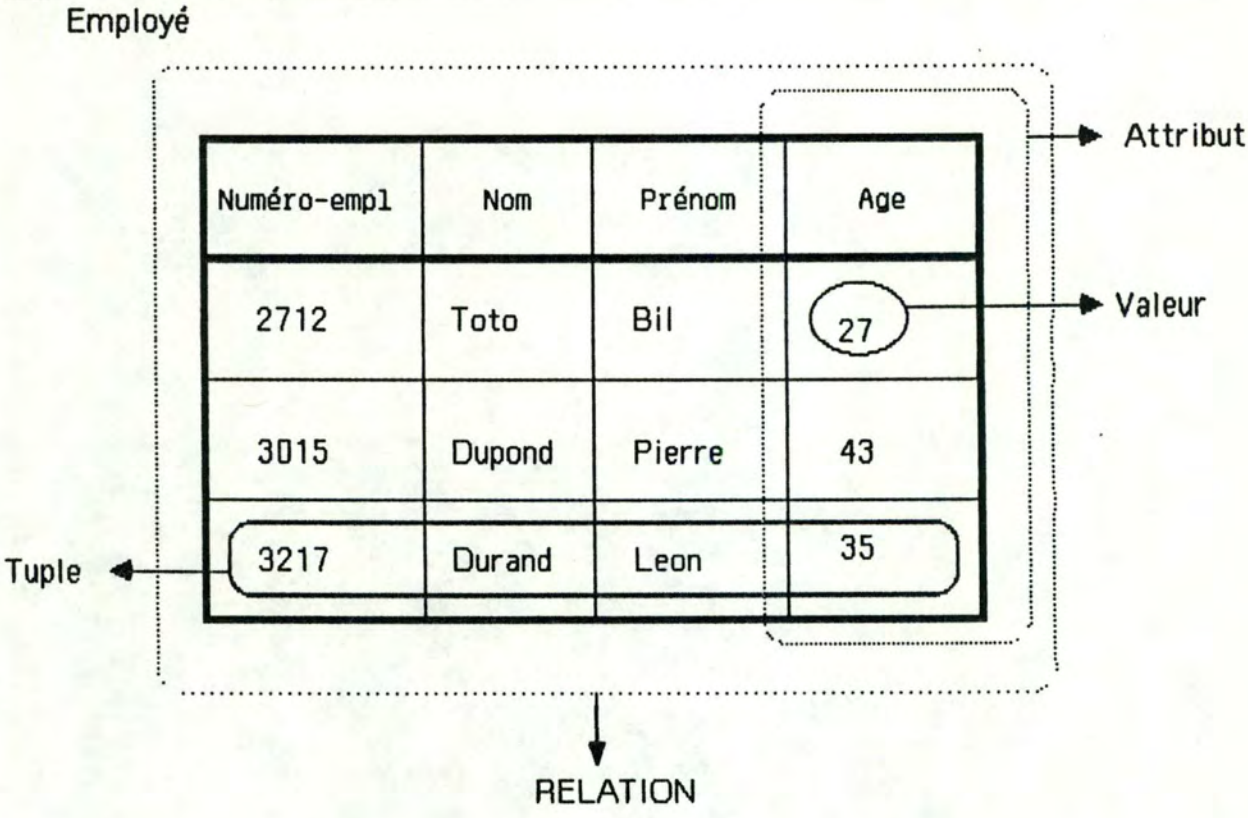
Toute table se conforme aux règles suivantes :

- Chaque case de la table contient une valeur.
- A l'intérieur d'une colonne, les valeurs des cases sont de même type.
- Chaque colonne porte un nom distinct.
- Chaque rangée est unique, il n'y a pas de copie.
- L'ordre des lignes et des colonnes n'a pas d'importance.

Cette approche est basée sur une théorie mathématique des ensembles qui reprend la terminologie suivante :

- Une table est appelée **relation.**
- Une rangée est appelée **un n-Tuple.**
- Une colonne est appelée **un attribut.**
- Les valeurs d'une colonne sont tirées d'un domaine.
- Le nombre de colonnes d'une table représente le degré de la relation.
- Le nombre de rangées d'une table représente la cardinalité de la relation.

exemple :



Notation mathématique :

Relation(X1,X2,...,Xn)

où Xi est un attribut auquel est attaché un domaine de valeurs DOM(Xi), pour i = 1, ..., n

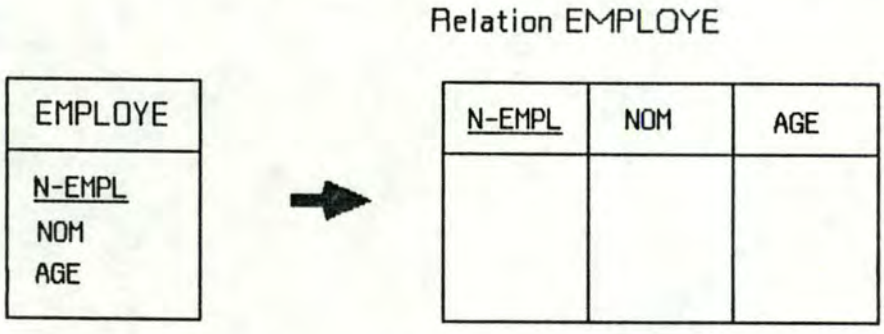
Nous trouvons également la notion de dépendance fonctionnelle :

Un goupe d'attribut A détermine fonctionnellement un groupe d'attributs B si pour une valeur de A correspond au plus une valeur de B.

Interprétation relationnelle du modèle entité-association

L'interprétation relationnelle la plus simple du modèle entité-association est la suivante :

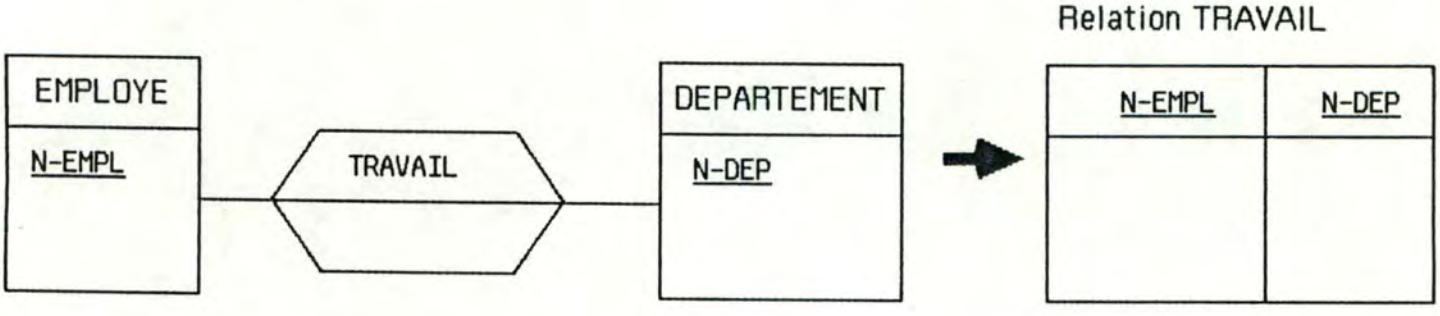
. Un type d'entité peut être représenté sous la forme d'une relation de la manière suivante :



où chaque rangée représente une entité de type EMPLOYE

Cette table est appelée une entité-relation ("entity-relation").

. De même, un type d'association peut être représenté sous forme d'une relation, mais comme tout type d'association est identifié par les identifiants des types d'entité sur lesquels le type d'association est défini, il faut reprendre ces identifiants comme identifiant de la relation créée.



Cette table est appelée association-relation ("relationship-relation").

Les formes normales

[er.01, de.11, rm.08, ot.14]

Nous nous contenterons de décrire ici les quelques formes normales qui interviendront dans la définition de la forme canonique.

Bien que liées au modèle relationnel, nous les présenterons en termes des concepts du modèle entité - association.

Pour être en première forme normale (1NF), un schéma e-a

- ne doit pas posséder d'attribut décomposable,
- tout attribut non identifiant dépend fonctionnellement de l'identifiant

Conséquence : un schéma e-a ne doit pas posséder d'attribut répétitif.

Pour être en seconde forme normale (2NF), un schéma e-a

- doit être en 1NF
- ne doit pas posséder d'attribut qui ne nécessite pas la totalité de l'identifiant pour l'identifier

Conséquence : un schéma e-a doit posséder une dépendance fonctionnelle totale de tout identifiant vers les attributs non identifiant.

Pour être en troisième forme normale (3NF), un schéma e-a

- doit être en 2NF
- ne doit pas posséder d'attribut non identifiant qui dépend fonctionnellement de l'identifiant par transitivité

Conséquence : Les attributs non identifiants doivent être mutuellement indépendants, c'est-à-dire qu'il ne doit pas y avoir de dépendance fonctionnelle entre attributs non identifiants.

Pour être en troisième forme normale forte (S3NF) [rm.04] , un schéma e-a

- doit être en 3NF
- tout déterminant est un candidat-identifiant

  . un déterminant est un (groupe d') attribut(s) sur lequel d'autres attributs dépendent fonctionnellement de façon totale.

  . un candidat-identifiant est un (groupe d') attribut(s) qui répond totalement à la définition d'identifiant.

2.0 Introduction

Ce chapitre est consacré à l'énoncé des règles que doit respecter un schéma entité - association afin de se trouver dans la forme canonique désirée.

La forme canonique que nous avons choisie doit permettre d'obtenir un schéma entité-association qui possède une redondance minimale, qui est complet et cohérent, c'est-à-dire que les propriétés de cette forme canonique sont telles qu'elles permettent d'exprimer au mieux la sémantique contenue dans un schéma entité - association. A cette fin, nous avons opté pour un certain nombre de règles définissant un schéma assez proche de la troisième forme normale de BOYCE et CODD connue aussi sous l'appellation de troisième forme normale forte (S3NF) (1).

Les caractéristiques de la forme canonique sont les suivantes :

- Unicité des noms de types d'entité, de types d'association, d'attributs, de rôles parmi tous les objets d'un schéma
- Au moins un identifiant pour tout type d'entité et d'association
- Pas d'attribut répétitif sauf dans le cas où l'attribut ne représente pas un concept autonome du schéma (2).
- Pas d'attribut ne nécessitant pas la totalité de l'identifiant pour l'identifier.
- Les attributs des types d'entité, types d'association doivent être mutuellement indépendants.
- Tout objet du schéma e-a doit posséder une spécification minimale (3).

---

(1) cf. Préambule
(2) cf. 2.3.2
(3) cf. 2.1

Nous apporterons une justification sur le choix de chacune des règles de la forme canonique au fur et à mesure de leur apparition dans les différents points de ce chapitre.Par contre,il nous semble intéressant de nous justifier, dès à présent, sur le fait que nous nous sommes écartés de la troisième forme normale forte.

Si nous nous sommes quelque peu écartés de cette forme normale, par l'ajout de certaines règles et par le relâchement d'autres, c'est parce que notre but principal, déjà énoncé, est d'offrir un certain nombre de règles qui permettent la meilleure expression possible de la sémantique contenue dans un schéma entité - association, sans se préoccuper de l'aspect implémentation future du schéma. Or, à notre avis, la troisième forme normale forte, comme toutes les formes normales, a tendance à favoriser cet aspect implémentation au détriment du but que nous recherchons.

En effet, nous avons constaté que la justification de l'introduction des différentes formes normales ont pour but essentiel d'obtenir un schéma relationnel qui permette d'éviter certains problèmes d'incohérence au niveau des mises à jour. D'autre part, lorsque le problème de la transformation d'un schéma e-a en schéma relationnel a été abordé, les études faites ont montré que certaines possibilités du modèle e-a pouvaient amener des propriétés structurelles indésirables pour la description d'une base de données. Ce qui a eu pour conséquence également l'introduction de ces formes normales qui avaient pour but la suppression de ces propriétés [er.01,de.11, rm.04,ot.14].

Ce chapitre est articulé autour des trois objectifs déjà mentionnés plus haut c'est-à-dire la redondance contrôlée, la complétude et la cohérence du schéma entité-association.

2.1 Complétude

2.1.0 Définition

Un schéma e-a sera dit complet si chaque élément qui fait partie du schéma, en notre possession, possède une spécification complète, c'est-à-dire que chaque concept repris dans ce schéma doit posséder, en fonction de son type, un ensemble de caractéristiques indispensables.

Pour cette raison, nous définirons, pour les différents concepts du modèle entité - association vus précédemment une spécification minimale, dans le sens où celle-ci doit comprendre un certain nombre d'informations indispensables, qui peuvent être complétées par d'autres éléments, spécifiés au point 1.1 du premier chapitre.

2.1.1 Contrainte portant sur les types d'entité

. Contenu minimal pour les types d'entité, comprenant les éléments suivants :

- un nom,
- une description qui devrait contenir la définition et la durée de vie du type d'entité
- au moins un identifiant,
- au moins un attribut le caractérisant.

L'essentiel de la spécification correspond à la définition constitutive attachée à ce type d'entité.
Il convient donc d'associer, au sein du contenu minimal, le nom et la définition de tout type d'entité.

Outre, ces deux informations indispensables, il convient également de préciser la durée de vie d'une occurrence d'un type d'entité.
En effet, une entité n'existe en tant que telle que durant la période où elle est considérée comme formant un tout, où il lui est conféré une existence autonome. Il est nécessaire de spécifier cette durée de vie qui vient compléter la définition constitutive.

Nous avons précédemment défini un type d'entité comme une chose concrète ou abstraite appartenant au réel perçu à propos de laquelle on veut enregistrer des informations, ce qui veut également dire qu'un type d'entité n'existe en tant que tel que par rapport à un individu ou groupe d'individus qui le considère comme un tout, lui confère une existence autonome et le distingue d'autres types d'entité et de son environnement.

Il convient donc d'associer à tout type d'entité au moins un identifiant qui est le mécanisme qui permet de faire la distinction entre entités d'un même type.
Eventuellement, il sera fait appel à un surrogate ce qui équivaut à définir un identifiant qui ne représente pas un concept réel du système d'information modélisé par le schéma e-a.

D'autre part, lorsque nous demandons que la spécification minimale de tout type d'entité contienne au moins une caractéristique de ce dernier c'est-à-dire qu'il soit spécifié au moins un attribut caractérisant celui-ci, cela vient du fait que nous avons défini le type d'entité comme étant une chose concrète ou abstraite à propos de laquelle on veut enregistrer des informations. Donc le fait de ne pas spécifier de caractéristique pour ce concept reviendrait à dire qu'un type d'entité a été créé sans pour autant qu'il soit question d'enregistrer des informations sur ce concept, autre que son existence, ce qui est peu vraisemblable.

2.1.2 Contrainte portant sur les attributs

. Présence obligatoire d'un contenu minimal pour un attribut constitué des éléments suivants :

- un nom,
- une description qui devrait contenir sa définition et sa durée de vie,
- son domaine de valeurs,
- simple ou répétitif,
- décomposable ou élémentaire,
- obligatoire ou facultatif,
- dérivable ou non,

Justifier la présence obligatoire d'un nom et d'une définition au sein du contenu minimal revient à reprendre les explications données au point précédent, pour ces mêmes informations mais cette fois-ci, en termes d'attribut.

Par contre, demander de compléter l'aspect descriptif par la présence de la spécification d'une durée de vie pour tout attribut, peut être justifié de la manière suivante :

Nous avons défini un attribut comme une qualité que peut posséder un type d'entité, type d'association. Cette qualité correspond à une prise de valeur à une période donnée.
Or, il est évident que celui-ci peut avoir une durée de vie indépendante du concept qu'il décrit, car durant sa durée de vie, il se peut que les propriétés d'un concept varient, dans le sens où certaines peuvent apparaître, d'autres s'évanouir.

En fait, lorsqu'un attribut est spécifié, en tant que qualité, propriété d'un autre objet, cet attribut représente l'ensemble des valeurs qu'il va pouvoir prendre. Cet ensemble de valeurs ou domaine de valeurs doit être spécifié dans le contenu minimal d'un tel objet car sans cette information, cela voudrait dire que nous avons spécifié une information que nous ne maîtrisons pas. Dans ce sens, le domaine de valeurs d'une caractéristique fait partie intégrante de sa définition.

Il nous semble fort important que lorsque le concepteur spécifie un attribut, il se rende compte de la redondance qu'il peut introduire dans le cas où cet attribut est dérivable. C'est pourquoi, nous demandons au concepteur de spécifier le fait qu'un attribut est est dérivable ou non à partir d'autres éléments de la structure d'informations. De plus, il conviendra d'exprimer la formule de dérivation de cet attribut sous forme de contrainte d'intégrité.

De plus, si cet attribut est décomposable, alors le contenu minimal de ce concept doit reprendre les noms de ses constituants. En effet, spécifier un attribut décomposable sans spécifier ses constituants n'a pas de sens.

2.1.3 Contrainte portant sur les types d'association

. Présence obligatoire d'un contenu minimal pour les types d'association, constitué des éléments suivants :

  . un nom,
  . une description qui devrait contenir sa définition et sa durée de vie,
  . pour chaque rôle joué au sein de ce type d'association :

    + le noms du rôle,
    + le type d'entité qui joue ce rôle,
    + la connectivité associée à ce rôle

Justifier la présence obligatoire d'un nom, d'une définition pour un type d'association, revient à reprendre les explications données sur ces mêmes informations pour le contenu minimal d'un type d'entité, mais en termes de type d'association.

D'autre part, comme un type d'association peut avoir une durée de vie inférieure à la durée de vie des types d'entité à partir desquels il est défini, il convient donc de spécifier la durée de vie de ce concept.

De plus, chaque type d'entité qui participe à un type d'association, y joue un ou plusieurs rôles. Il est donc important de faire apparaître les rôles joués par ces objets au sein du contenu minimal au même titre que leur nom, ainsi que la connectivité associée à ces différents noms de rôles.

Par définition d'un type d'association, il convient de spécifier les types d'entité mis en correspondance par ce type d'association.

2.2 Redondance controlée

2.2.0 Définition

Nous définirons la redondance comme une information non nécessaire, se trouvant dans le schéma e-a, car celle-ci réexprime une information déjà présente dans ce schéma.

Les règles suivantes, regroupées en fonction des constituants du schéma e-a, ont pour but de minimiser et contrôler cette redondance.

2.2.1 Contraintes portant sur les types d'entité

. Pas de noms de type d'entité différents, pour un même contenu (synonymie) :

un type d'entité ne peut être présent, dans le schéma e-a, à plusieurs reprises, en portant des noms différents.

Si dans le schéma entité-association, nous trouvons deux types d'entité portant des noms différents, mais qui ont même spécification, alors cela représente un signe d'erreur de modélisation. En effet, cela signifierait que le concepteur modélise à deux reprises une chose abstraite ou concrète mais sous des appellations différentes.Ce qui signifie qu'un de ces deux concepts n'apporte aucune information complémentaire et est donc redondant et doit être supprimé.

. Tout type d'entité doit posséder un identifiant minimal (3NF) :

c'est-à-dire qu'il ne doit pas exister de dépendance fonctionnelle entre les éléments constituant l'identifiant. (1)

(1) cf. Annexe A.4 : Obtention d'un identifiant minimal

Comme nous l'avons signalé pour la complétude, tout type d'entité doit posséder au moins un identifiant, mais en plus, il doit être minimal. En effet, s'il existe une dépendance fonctionnelle entre éléments de l'identifiant, le(s) élément(s) déterminé(s) de cette dépendance peu(ven)t être supprimé(s) de l'identifiant sans pour autant détruire le mécanisme d'identification de l'identifiant.
Ce qui veut dire que les éléments écartés de l'identifiant ne constitueraient qu'une information inutile au sein de l'identifiant (donc redondance).

2.2.2 Contraintes portant sur les attributs

. Pas d'attributs portant des noms différents, pour un même contenu (synonymie) :

un attribut ne peut être présent, dans le schéma, à plusieurs reprises, en portant des noms différents.

Justifier l'interdiction de synonymie au sein des noms d'attributs revient à reprendre la justification énoncée pour ce point, en ce qui concerne les types d'entité, mais ici en termes d'attributs.

2.2.3 Contraintes portant sur les types d'association

. Pas de types d'association portant des noms différents pour un même contenu (synonymie) :

un type d'association ne peut être présent, dans le schéma e-a, à plusieurs reprises, en portant des noms différents.

. Pas de type d'association sémantiquement redondant avec la composition de plusieurs autres types d'association.

Lorsque nous interdisons les types d' association sémantiquement redondant, c'est parce que ces types d'association sont formellement dérivables d'autres types d'association. Ce qui constitue une information n'apportant rien et qui peut donc être supprimée.

. L'identifiant d'un type d'association doit être minimal (3NF) :

c'est-à-dire qu'il ne peut exister de dépendance fonctionnelle entre les éléments constituant l'identifiant du type d'association, et ce type d'association ne doit plus être décomposable. (1)

Comme nous l'avons déjà signalé en termes de complétude, tout type d'association doit posséder au moins un identifiant, mais en plus, il doit être minimal.

En effet, s'il existe une dépendance fonctionnelle entre éléments de l'identifiant, le(s) élément(s) déterminé(s) de cette dépendance peu(ven)t être supprimé(s) de l'identifiant sans pour autant détruire le mécanisme d'identification. Ce qui veut dire que les éléments écartés de l'identifiant ne constituaient qu'une information inutile au sein de cet identifiant. De plus, nous signalons que ce type d' association ne doit plus être décomposable. En effet, comme il l'est signalé au point 2.3.3, si un type d'association ne possède pas un degré minimal, il sera décomposé, ce qui aura pour effet d'apporter des modifications sur l'identifiant de cet objet. Par contre, une fois que le degré minimal est atteint pour cet objet, la seule manière de s'assurer de la présence d'un identifiant minimal est de s'assurer de l'absence de toute dependance fonctionnelle.

(1) cf. Annexe A.4 : Obtention d'un identifiant minimal

2.2.4 Contraintes portant sur les contraintes d'intégrité

. Pas de contrainte d'intégrité définissant un domaine de valeurs ou un ensemble d'occurrences identique à celui défini par plusieurs autres contraintes (l'ensemble des contraintes d'intégrité étant reliées par un ET implicite):

En effet, dans ce cas, celle-ci n'apporte aucune sémantique supplémentaire, et peut donc être supprimée.

. Pas de dépendance fonctionnelle transitive :

La relation de dépendance fonctionnelle possède la propriété de transitivité. Il n'est donc pas nécessaire de spécifier une dépendance fonctionnelle qui est transitive, car elle n'apporte aucune sémantique.

2.3 Cohérence

2.3.0 Définition

Un schéma e-a sera dit cohérent s'il n'existe pas de contradiction au sein des définitions des concepts repris dans le schéma e-a, complèté par l'ensemble des contraintes d'intégrité définies par le concepteur, que cette contradiction se situe au niveau des spécifications de ces concepts, ou par rapport à la définition donnée au premier chapître.

2.3.1 Contraintes portant sur les types d'entité

. Unicité des noms de type d'entité (homonymie) :

deux types d'entité différents ne peuvent porter le même nom.

Ayant déjà justifié notre choix concernant l'unicité des noms de types d'entité, nous nous contenterons d'apporter quelques explications sur la présence des seconde et troisième règles.

. Un type d'entité ne peut avoir comme sous-type un type d'entité étant d'un niveau supérieur ou égal dans la hiérarchie des sous-types (incompatibilité de sous-types).

En effet, si cela était permis, cela voudrait dire que nous avons, vu que le sous-type est transitif, qu'un type d'entité A peut être à la fois "super-type" et "sous-type" d'un type d'entité B, ce qui est incohérent.

Exemple :

Hiérarchie de " sous-type"

. Restriction "pratique" sur les identifiants d'un type d'entité :

> Il serait raisonnable d'admettre que lorsqu'un ou plusieurs noms de rôle apparaissent au sein de l'identifiant d'un type d'entité, la connectivité associée doit être égale à (0-1) ou (1-1) au sein d'un type d'association de degré 2 [er.14]

2.3.2 Contraintes portant sur les attributs

. Unicité des noms d'attribut (homonymie) :

deux attributs différents ne peuvent porter le même nom.

l'unicité des noms d'attribut n'est qu'une reprise d'une contrainte déjà spécifiée au niveau de la forme générale, c'est pourquoi nous nous contenterons d'apporter uniquement des explications sur les deux autres contraintes.

. Pas d'imbrication d'attributs :

c'est-à-dire pas d'attribut décomposable ou d'attribut élémentaire entrant de plusieurs manières dans une structure.

ex : un attribut qui apparaîtrait à la fois comme élément d'un attribut décomposable et comme simple attribut au sein du concept qu'il décrit.

. A priori, élimination de tout attribut élémentaire ou décomposable répétitif, sauf dans le cas où il ne représenterait pas un concept autonome : (1)

En effet, la présence d'un tel attribut est souvent le signe de la présence d'un type d'entité, qui n'a pas été détecté, au sein du type d'entité contenant cet attribut. En éliminant ce type d'attribut, nous permettons une meilleure expression de la sémantique du schéma entité - association.

On évitera cependant de considérer que l'existence d'un attribut répétitif au sein d'un type d'entité ou d'un type d'association, doit obligatoirement entraîner la désagrégation (2) de ce type d'entité ou d'association. En effet, un attribut répétitif ne représente pas nécessairement un concept autonome du réel perçu.

---

(1) cf. Annexe A.2 : Suppression d'un attribut répétitif

(2) désagrégation : Lorsqu'un attribut représente un concept autonome du système d'informations, il sera supprimé du type d'entité ou d'association qu'il décrivait et on créera soit un nouveau type d'entité, soit un nouveau type d'association défini sur le type d'entité initial

---

. Pas de dépendance fonctionnelle partielle par rapport à l'identifiant (1) (2).

Tout attribut doit nécessiter l'entièreté de la clé identifiante pour l'identifier. En effet, si au sein d'un type d'entité, par exemple, nous trouvons un attribut qui ne serait identifié que par une partie de l'identifiant, cela serait l'indice d'une erreur de modélisation de la part du concepteur, dans le sens où le type d'entité représente ici deux concepts qui possèdent une existence autonome. Et donc, afin d'éviter une telle erreur de modélisation, nous interdisons ce fait.

. Tous les attributs d'un type d'entité ou d'association doivent être mutuellement indépendants : (1)

c'est-à-dire qu'il ne doit pas exister de dépendance fonctionnelle autre que celle de l'identifiant, vis à vis des autres attributs.

En effet, la présence d'une dépendance fonctionnelle A entre deux attributs non identifiants, rend la dépendance fonctionnelle qui existe entre l'identifiant et l'attribut déterminé de la dépendance A, redondante par transitivité de dépendance fonctionnelle. Or, comme parmi les règles de contrôle de la redondance, nous trouvons une règle qui interdit ce genre de dépendance. D'autre part, cette dépendance transitive ne peut être éliminée, par définition d'un identifiant.

En conclusion, nous pouvons dire qu'admettre des dépendances fonctionnelles entre attributs non identifiant, revient à contredire une règle de contrôle de redondance et donc à ne plus respecter la forme canonique.

---

(1) cf. Préambule : Formes normales

(2) cf. Annexe A.3 : Attribut(s) ne dépendant pas fonctionnellement de la totalité de l'identifiant

2.3.3 Contraintes portant sur les types d'association

. Unicité des noms de type d'association (homonymie) :

deux types d'association différents ne peuvent porter le même nom.

La justification des deux premières règles est tout à fait similaire aux explications proposées pour les types d'entité, mais en termes de types d'association.

. L'identifiant d'un type d'association ne peut être constitué que des noms de rôle des types d'entité participant à celui-ci.

REMARQUE :

Il s'agit d'une contrainte de modélisation.

. Pas de type d'association possédant une cardinalité et une connectivité contradictoires :

c'est-à-dire que si l'on a pour un type d'association, une cardinalité I, et une connectivité 1-J, pour un type d'entité participant à ce type d'association, alors il y aura contradiction si I<J.
Ce qui revient à dire que la cardinalité spécifiée ne doit pas être en contradiction avec celle qui peut être déduite de la cardinalité et de la connectivité des types d'entité participant à ce type d'association.

En effet, la relation mathématique suivante doit toujours être vérifiée, dans le cas où toutes les cardinalités en tant que nombre maximun d'occurrences du type d'entité sont connues :

min(cardinalité(entité) * min-connectivité(rôle))

<= Cardinalité(association) <=

max(cardinalité(entité) * max-connectivité(rôle))

La cardinalité d'un type d'association doit être au plus égale au max du produit de la cardinalité et de la multiplicité des types d'entité qui jouent un rôle au sein de ce type d'association.

---

(1) cf. chapitre 1

. Le degré d'un type d'association doit être minimal : (1)

Lorsque le degré d'un type d'association n'est pas minimal, cela veut dire que plusieurs concepts ont été modélisés par cet objet. Il convient donc de décomposer ce type d'association afin de faire apparaître explicitement ces concepts. Ceci permettrait une meilleure expression de la sémantique du schéma.

Si le type d'association est décomposable, c'est-à-dire qu'il est en accord avec une des conditions suivantes, il sera décomposé selon les règles définies en annexe.

Un type d'association est décomposable si son degré est supérieur à 2 et si la condition suivante est vérifiée :

S'il existe une dépendance fonctionnelle d'un seul nom de rôle vers un autre nom de rôle.

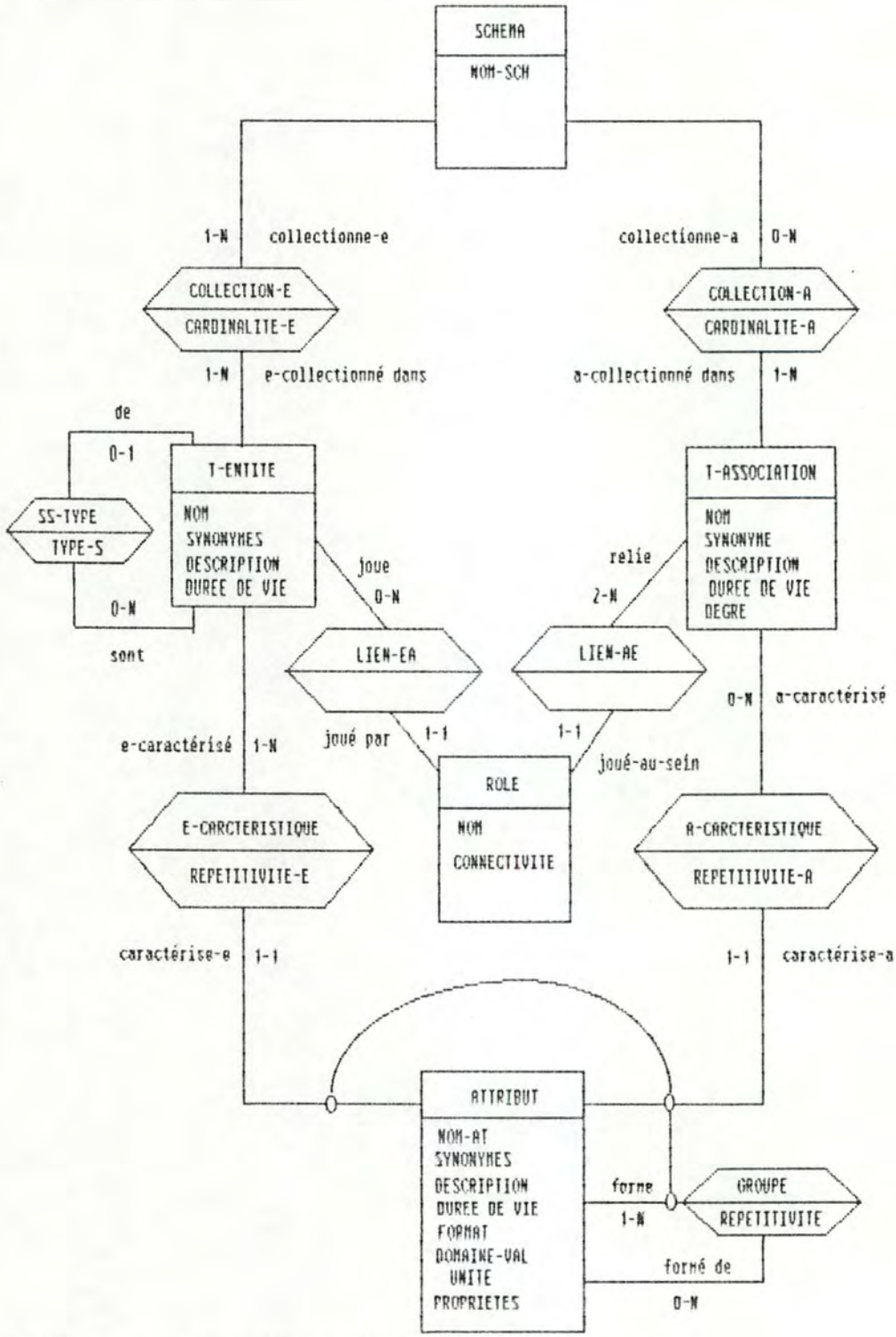
(1) cf. Annexe A.1 : Décomposition de type d'association

2.3.4 Contraintes portant sur les contraintes d'intégrité

. Pas de contrainte d'intégrité incompatible

2.4 Schéma entité-association du modèle entité-association canonique

2.4.1 Le modèle entité-association



Remarque : - *Le domaine de valeur de "propriété" est défini en extension par :*

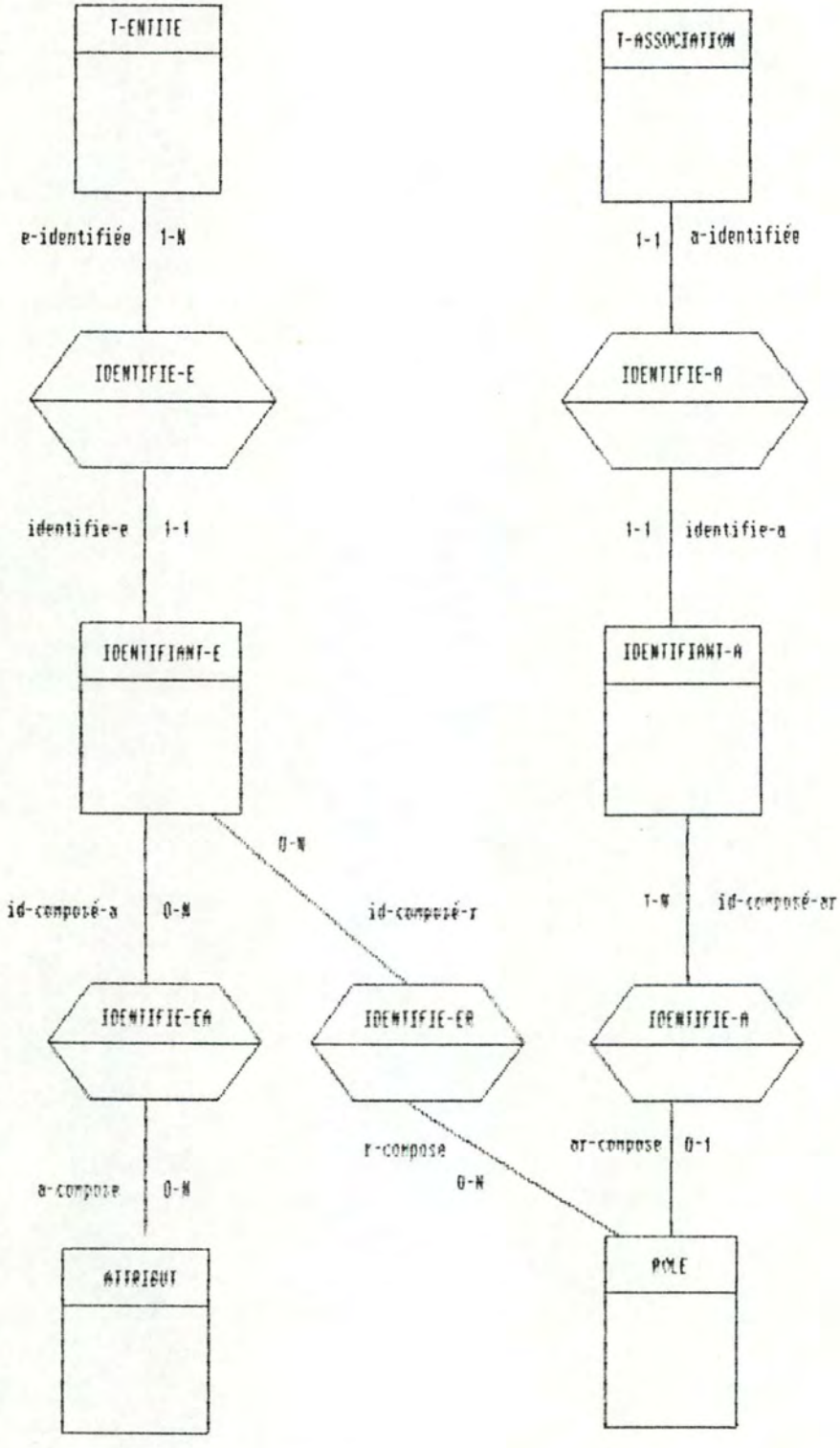
simple ou répétitif, élémentaire ou décomposable, obligatoire ou facultatif, dérivable ou non

- 0———0 *représente une contrainte d'exclusion de rôles*

Contraintes : - Le domaine de valeur de "type-s" est défini en extension par :

faible ou fort

2.4.2 Le modèle d'identification



Remarque : un identifiant d'une entité doit être composé d'au moins un attribut ou d'un rôle

2.4.3 Le modèle des dépendances fonctionnelles

Pour un schéma e-a canonique, seules les dépendances fonctionnelles entre rôles et/ou groupes de rôles sont permises. Le schéma qui décrit les dépendances fonctionnelles permises est identique à celui de la page 31.

CHAPITRE 3 REPRESENTATION D'UN SCHEMA ENTITE-ASSOCIATION EN DSL

3.0 Introduction

Après avoir introduit progressivement la forme canonique finale vers laquelle doit tendre tout schéma e - a, nous allons nous fixer sur le choix d'un langage de spécification qui permette la formalisation de l'expression d'un schéma conceptuel et qui, par conséquent, permette l'étude du système de pilotage.

Au cours de ce chapitre,nous nous placerons dans le cadre de l'atelier-logiciel IDA, et nous montrerons de quelle manière peut être représenté un schéma entité-association canonique au moyen du langage de spécification DSL [ot.31], développé dans cet atelier.

En vue d'atteindre cet objectif, nous présenterons d'une part, la correspondance qui existe entre les concepts du modèle e-a et leur représentation en DSL, et d'autre part, nous spécifierons les contraintes qui devront compléter la définition de la forme canonique du fait de l'utilisation de DSL.

Préalablement, nous énumèrerons les caractéristiques principales du langage DSL.

3.1 Caractéristiques générales du langage DSL

Le langage de spécification DSL, développé dans le cadre de l'atelier-logiciel IDA a été conçu comme un outil de représentation des spécifications fonctionnelles. Dès lors, il n'est pas étonnant qu'il couvre des domaines plus vastes que le seul aspect "schéma conceptuel d'informations". Cependant, dans le cadre de notre travail, nous nous limiterons à présenter les caractéristiques générales de ce langage sur base de cet unique aspect.

Le langage DSL est basé principalement sur les concepts d'objet et de relation qui permet d'établir des liens entre les différents objets.
Pour chacun de ces concepts, il existe un certain nombre précis de type :

- les différents types d'objet du langage DSL que l'on retrouve sous l'aspect qui nous intéresse sont :

  . ELEMENT, GROUP, ENTITY, RELATION, ROLE, SET

- d'autre part, les différents types de relation qui peuvent exister entre ces objets et que l'on peut retrouver autour de cet aspect sont principalement :

  . SUBTYPE, CONSIST, VALUE, IDENTIFIES, DETERMINES, FORMAT, RELATES, CONNECTIVITY, CARDINALITY

En fait, si l'on réexamine le schéma de la page 55, il est assez aisé de remarquer que le langage DSL est basé sur le modèle de structuration des données entité - association où tout type d'objet cité précédemment représente un type d'entité de ce schéma et où tout type de relation représente un type d'association ou un attribut de ce schéma.

A partir de là, il sera très facile de comprendre la correspondance qui existe entre les concepts du modèle entité - association et leur représentation en DSL qui vous est présentée au point suivant.

3.2 Correspondance entre le modèle entité - association et sa représentation en DSL

3.2.1 Tableau de correspondance

Le tableau qui suit, illustre la correspondance qui existe entre les éléments constitutifs du modèle entité - association et les concepts de base du langage DSL. La description de ces éléments dans le langage DSL est donnée dans l'annexe B.

CONCEPTS DU MODELE ENTITE-ASSOCIATION	REPRESENTATION EN DSL DES CONCEPTS	
	Representation	Nature DSL du concept
Type d'entité	ENTITY	objet
Type d'association	RELATIONSHIP	objet
Attribut		
élémentaire	ELEMENT	objet
décomposable	GROUP	objet
Description d'un élément	DESCRIPTION	relation
Durée de vie	DESCRIPTION	relation
Identifiant d'un T. E., T. A.	IDENTIFIES	relation
Rôle d'un T.E.	ROLE IS	objet
Connectivité	CONNECTIVITY	relation
Cardinalité	CARDINALITY	relation
Domaine d'un attribut	SAME DOMAIN AS VALUE IS	relation relation
Format d'un attribut	FORMAT IS	relation
Table de codification	CODE ... MEANS	relation
Dépendance fonctionnelle	FUNCTIONNALY ...	relation
Dérivabilité d'un attribut	KEYWORDS ...	
Autres contraintes d'intégrité	cf. point 3.2.2	

Remarque :

A) Pour définir, le déterminant ou le déterminé d'une dépendance fonctionnelle ou un identifiant, lorsque ce concept est constitué d'une combinaison d'objets "element","group" ou "role" indépendante de leur structure sémantique, DSL offre un type d'objet appelé "AGGREGATE".

B) De la même manière, DSL offre un type d'objet "SYSTEM-PARAMETER" qui sera fort utile pour définir des concepts tels que la répétitivité, la cardinalité et la connectivité car ce type d'objet permet d'exprimer le caractère variable des valeurs que peuvent prendre ces différents concepts.

Règle de codification du "system-parameter"

Comme tous les autres objets du langage DSL, l'objet "system-parameter" doit porter un nom qui le distingue de tout autre objet de la base de données IDA.

La spécification d'un nom pour ce type d'objet doit respecter la règle suivante :

- Lorsque l'utilisateur désire spécifier une valeur minimale et une valeur maximale via le nom d'un objet "sys-par", la syntaxe que devra respecter le nom sera :

  sp-<min>-<max>-<identificateur>

  ex: pour exprimer une connectivité 1-N, on peut spécifier un "sys-par" de nom :

  sp-1-N-toto

L'utilisation de <identificateur> est nécessaire pour distinguer plusieurs objets "system-parameter" qui peuvent avoir mêmes valeurs minimale et maximale.

- Pour tout autre cas, le choix d'un nom est laissé à l'appréciation de l'utilisateur.

C) A l'heure actuelle, le langage DSL ne possède pas de relation spécialement prévue afin d'exprimer le fait qu'un attribut est dérivable ou non. Or, comme cette caractéristique est un élément obligatoire de la spécification minimale de tout attribut (1), il convient de lui définir un mode de représentation, au moyen des outils à notre disposition.

Ce mode de représentation sera donc :

. Pour exprimer la dérivabilité d'un attribut, nous aurons recours à la relation suivante :

KEYWORDS derivable; (2)

D) Pour un certain nombre de caractéristiques telles que la répetitivité d'un attribut au sein de l'objet qu'il décrit, la dérivabilité d'un attribut, il est possible de spécifier des valeurs par défaut, c'est-à-dire que l'absence de toute spécification sera interprétée de la manière suivante :

l'absence de spécification pour la dérivabilité aura pour conséquence de considérer l'objet comme non dérivable.

l'absence de spécification pour la répétitivité aura pour conséquence de considérer la répétitivité égale à 1.

E) On considèrera qu'à tout objet relation, l'identifiant minimal obtenu pour celui-ci, lui est associé. Aucune relation "IDENTIFY BY ..." ne doit donc être spécifiée pour un objet "relation".

---

(1) cf. chapitre 2, pt 2.1.2

(2) Il est également possible d'utiliser la relation "ATTRIBUTE IS" pour spécifier le caractère dérivable d'un attribut : "ATTRIBUTE derivability YES;", comme le demande Jean Decrane dans le cadre d'un système qui ne possède pas la relation "KEYWORDS".

3.2.2 Représentation des contraintes d'intégrité

Comme nous l'avons déjà mentionné, il est indispensable de permettre à tout utilisateur de spécifier ses contraintes d'intégrité. Mais, si nous lui offrons cette possibilité, cela veut également dire que nous devons être capable de lui fournir un outil d'analyse des contraintes afin de s'assurer de la cohérence et de la non-redondance de ces dernières.

Or, si nous laissons le libre choix d'une syntaxe pour spécifier les contraintes , à l'utilisateur, alors l'analyse de ces dernières devient impossible.

C'est donc dans le but de rendre possible la vérification de la cohérence et de la non-redondance que nous nous proposons de présenter un langage de spécification des contraintes d'intégrité.

Pour effectuer notre choix parmi les langages étudiés [er.11,er.12,er.13], nous nous sommes basés sur un certain nombre de critères :

a) Ce langage de spécification doit être assez général pour permettre la prise en compte d'au moins l'ensemble des classes de contraintes d'intégrité les plus usuelles définies au point 1.2 du chapitre 1.

b) Ce langage doit se baser uniquement sur les concepts du modèle entité-association, et doit être indépendant de tout choix d'implémentation futur, c'est-à-dire de tout SGBD existant.

c) Cependant, ce langage doit être conçu de telle manière qu'il permette une translation assez aisée des contraintes spécifiées en termes de SGBD existants.
En effet, si le fait d'introduire dès le niveau conceptuel, les contraintes d'intégrité, ne peut se concrétiser par une traduction "automatique" de celles-ci vers un SGBD existant, nous perdons tout l'intérêt de cette intégration à ce niveau.

choix qui s'est finalement porté sur DESPATH de W. Roesner [er.13].

Afin de justifier le choix que nous avons effectué, nous allons vous présenter une étude comparative des langages DESPATH [er.13] et ERROL [er.11].

Cette étude s'effectuera en deux temps avec d'une part, la présentation des concepts de base, de la syntaxe de ces différents langages, et d'autre part, en guise d'illustration, nous reprendrons les exemples de contraintes d'intégrité données au point 1.2 du chapitre 1 que nous traduirons dans ces deux langages avant d'en tirer des conclusions.

Remarque :

Vous pourrez rapidement remarquer que malgré le premier critère que nous nous nous sommes fixé, aucun des langages étudié n'aborde l'aspect contrainte d'intégrité dynamique. Toutefois, nous ne vous proposerons pas d'extension à ces langages pour prendre en compte cet aspect car l'introduction de la notion de temps au sein de ces langages, nous obligerait à introduire le concept de base de données historique, ce qui sortirait de notre propos.

3.2.2.1 Syntaxe des langages

Au cours de ce paragraphe, nous nous contenterons de décrire les caractéristiques générales du langage ERROL et DESPATH. Nous laissons le soin au lecteur de consulter les références [er.11] et [er.13] pour avoir un exposé exhaustif de la syntaxe d'ERROL et de DESPATH.

Concepts de base pour ERROL

- Le nom des types d'entité, des rôles et des attributs sont considérés comme des identifiants.
- L'apparition d'un nom de rôle sera toujours préfixée par une apostrophe.
- Un type d'association sera représenté par une paraphrase contenant au moins un type d'entité et le rôle joué par ce type d'entité au sein de ce type d'association.

  ex : CLIENT 'passe COMMANDE (pour PASSATION)

- L'expression des contraintes d'intégrité se font sous forme de requetes au moyen de primitives tels que :

  GET, SET

  d'opérateurs logiques de combinaison de critères tels que :

  AND, OR

  d'opérateurs de qualification qui permettent de délimiter les ensembles de valeurs définis à l'aide des primitives :

  TIS, < , ...

  ex : Tout produit commandé doit être repris dans les différents catalogues.

  se traduira par :

  GET produit TIS NOT CONTAINS IN SET catalogue;

Concepts de base pour DESPATH

- Les noms des types d'entité, des rôles et des attributs sont considérés comme des identifiants.
- L'apparition d'un nom de rôle sera toujours préfixée et postfixée par une apostrophe.
- Ce langage est basé sur quelques primitives qui peuvent être imbriquées,

  ex : FIND, SELECT

ainsi que sur des opérateurs de qualification qui permettent de délimiter les ensembles de valeurs définis à l'aide des primitives précédentes.

ex : WITH, THAT, WHICH, ...

et des oprérateurs logique qui permettent de combiner plusieurs critères.

ex : AND, OR

3.2.2.2 Illustration

A ce niveau nous allons traduire les différentes contraintes exprimées, comme illustration, au point 1.2.

a) Une commande ne peut exister que si elle porte sur au moins un produit présent dans le catalogue.

```
DESPATH :
        COUNT ( FIND COMMANDE
                     THAT 'contient'
                          PRODUIT
                          WHICH ¬'est présent dans'
                                CATALOGUE) = 0

ERROL :
        GET COMMANDE TIS 'contient COUNT SET PRODUIT
            TIS 'est présent dans CATALOGUE = 0
```

b) Tout produit doit être de type électro-ménager, hifi, peinture, bois/verre, électricité.

```
DESPATH :
        COUNT(FIND PRODUIT
                   ( WHICH ¬'est de type em'
                           ELECTRO-MENAGER )
                    AND
                   ( WHICH ¬'est de type h'
                           HIFI )
                    AND
                   ( WHICH ¬'est de type el'
                           EECTRICITE )
                    AND
                   ( WHICH ¬'est de type p'
                           PEINTURE )
                    AND
                   ( WHICH ¬'est de type bv'
                           BOIS/VERRE ) ) = 0

ERROL :
        GET PRODUIT TIS NOT 'est de type em
                        AND NOT 'est de type h
                        AND NOT 'est de type el
                        AND NOT 'est de type p
                        AND NOT 'est de type bv
```

c) Si un produit est de type electro-ménager, il ne peut être de type bois/verre, peinture, électricité

```
DESPATH :
          COUNT( FIND PRODUIT
                      ( WHICH 'est de typz em'
                              EECTRO-MENAGER )
                        AND
                      ( ( WHICH 'est de type bv'
                                BOIS/VERRE )
                          OR
                        ( WHICH 'est de type p'
                                PEINTURE )
                          OR
                        ( WHICH 'est de type el'
                                ELECTRICITE ) ) ) = 0

  ERROL :
          GET PRODUIT TIS 'est de type em
                          ELECTRO-MENAGER
                       AND 'est de type bv
                          BOIS/VERRE
          GET PRODUIT TIS 'est de type em
                          ELECTRO-MENAGER
                       AND 'est de type p
                          PEINTURE
          GET PRODUIT TIS 'est de type em
                          ELECTRO-MENAGER
                       AND 'est de type el
                          ELECTRICITE
```

d) Si un produit est de type hifi, il est également de type électro-menager.

```
DESPATH :
          COUNT ( FIND PRODUIT
                      ( WHICH 'est de type'
                              HIFI )
                        AND
                      ( WHICH ¬'est de type em'
                                ELECTRO-MENAGER) ) = 0

  ERROL :
          GET PRODUIT TIS 'est de type h HIFI
                      AND NOT 'est de type em
                              ELECTRO-MENAGER
```

e) Le montant d'une commande doit toujours être > 0

```
DESPATH :
          COUNT (FIND COMMANDE WITH MONTANT <= 0 ) = 0

  ERROL :
          GET COMMANDE TIS 'HAVING "montant <= 0"
```

f) Numéro de client dépend fonctionnellment de l'adresse du client.

Déjà exprimable en DSL

g) Si "q-stock" = 0, alors "date-app" doit être présente.

```
DESPATH :
          COUNT ( FIND PRODUIT WITH (Q-STOCK = 0) &
                                    (DATE-APP = nulle)
                ) = 0
  ERROL :
          GET PRODUIT TIS 'HAVING "Q-STOCK = 0"
                              AND "DATE-APP = 0"
```

h) Une occurrence de l'objet "relation" "attente" ne peut exister que si elle porte sur une commande dont au moins un produit a une quantité en stock égale à 0.

```
DESPATH :
          COUNT ( FIND ATTENTE
                       OF COMMANDE
                          THAT 'contient'
                               PRODUIT
                                  WITH Q-STOCK = 0) = 0

  ERROL :
          GET COMMANDE TIS 'contient COUNT(PRODUIT
              TIS 'HAVING "Q-STOCK = 0" ) = 0
                        AND 'en attente COM-DIFFEREE
```

i) Les occurrences de COMMANDE qui participent à une occurrence de ATTENTE doivent être différentes des occurrences de COMMANDE qui participent à une occurrence de FACT-COM.

```
DESPATH :
          COUNT ( FIND COMMANDE
                       THAT 'en attente'
                            COM-DIFFEREE
                    AND
                       THAT 'donne lieu à'
                            FACTURE ) ) = 0
  ERROL :
          GET COMMANDE TIS 'en attente
                        AND 'donne lieu à
```

j) L'ensemble des produits qui jouent le rôle "est de type h" est inclus dans l'ensemble des produits qui jouent le rôle "est de type em".

```
DESPATH :
          COUNT ( FIND PRODUIT
                       ( THAT 'est de type h'
                              HIFI )
                        AND
                       ( THAT ¬'est de type em'
                              ELECTRO-MENAGER ) ) = 0

  ERROL :
          GET PRODUIT TIS 'est de type h
                       AND NOT 'est de type em
```

k) L'ensemble des commandes passées par les clients est égal à l'ensemble des commandes constituées par l'union des commandes en attente et des commandes facturées.

```
DESPATH :
          COUNT ( FIND COMMANDE
                       ( WIHCH 'passée par'
                               CLIENT )
                        AND
                       ( THAT ¬'donne lieu à'
                                 FACTURE )
                        AND
                       ( THAT  ¬'en attente'
                                 COM-DIFFEREE ) ) = 0
  ERROL :
          GET COMMANDE TIS 'passée par
                        AND NOT 'donne lieu à
                        AND NOT 'en attente
```

Remarque :

Dans DESPATH, seul le mécanisme du COUNT permet d'exprimer des contraintes qui portent sur des entités. Seul les ensembles d'attributs peuvent être comparés.

Conclusion

Après avoir traduit les contraintes du point 1.2 du chapitre 1, il nous est apparu que DESPATH avait une structure plus proche du langage naturel. D'autre part, la traduction des contraintes dans le formalisme proposé par DESPATH s'avère beaucoup plus systématique que pour ERROL et le résultat obtenu est beaucoup plus lisible.
Dans le cas où un schéma e-a doit être transformé en un schéma relationnel, il s'avère que le langage DESPATH est le plus intéressant dans la mesure où il est fort proche du langage SQL qui tend à devenir un standard pour le traitement des bases de données.

De plus, le mécanisme du COUNT qu'offre DESPATH et non présent dans ERROL permet l'expression réelle de contraintes d'intégrité alors que ERROL ne permet que l'expression de requête dont le résultat doit être analysé afin de s'assurer du respect ou du non-respect de la contrainte ainsi exprimée.

3.2.2.3 Proposition d'extension du langage DSL

Toute contrainte d'intégrité a pour but de restreindre le domaine de valeurs d'un objet. Nous dirons donc qu'á toute contrainte d'intgrité est associé un domaine contraint.

D'autre part, tout domaine contraint est contraint soit par un ou plusieurs objets différents, soit par une constante.

Afin de représenter les différents concepts liés aux contraintes d'intégrité que nous venons de définir, nous nous proposons d'introduire un nouveau type d'objet : "CONSTRAINT", auquel peuvent être associées les relations suivantes :

```
DEFINE CONSTRAINT constraint-name;

    CONSTRAINED DOMAIN IS  { element-name  } ;
                           { group-name    }
                           { entity-name   }
                           { relation-name }

    RESTRICTING DOMAIN IS  ( { element-name  } [,] )
                             { group-name    }
                             { entity-name   }
                             { relation-name }
                             { < string >    }
                             { <integer >    }

    CONSTRAINT-TEXT IS <text> ;
```

Dans la relation " CONSTRAINT-TEXT ...", le texte doit respecter la syntaxe définie au point E.2 précédent.

3.3 Restrictions aux possibilités offertes par DSL

Le fait d'employer le langage de spécification DSL, nous oblige á ajouter un certain nombre de contraintes á celles déjá exprimées lors de la définition de la forme canonique, afin que celle-ci soit respectée, car DSl offre de nombreuses possibilités de spécification qui ne répondent pas aux définitions données au cours du premier chapitre.

REMARQUE :

Outre les contraintes propres au langage de spécification DSL, il existe également un certain nombre de contraintes supplémentaires associées á l'utilisation de l'extracteur de la base de données IDA. Celles-ci vous sont présentées en annexe D, ainsi qu'une spécification générale de cet extracteur.

3.3.1 Contraintes portant sur les objets "entity"

. Pas d'objet "entity" possédant une relation "IDENTIFIED BY" qui porte sur :

  . 1 ou plusieurs objets "element" et/ou "group" qui ne participent pas á une relation "CONTAINED IN" portant sur cet objet "entity".

et/ou . 1 ou plusieurs objets "role" qui ne sont pas joués par des objets "entity" lié via un objet "relation" á cet objet "entity"

Tout identifiant d'un objet "entity" sera considéré comme incohérent s'il ne respecte pas la définition faite dans la forme générale (chapitre 1).

. La spécification d'un objet "entity" ne peut contenir le nom d'un objet "element/group" inexistant dans le schéma.

3.3.2 Contraintes portant sur les objets "element/group"

. Pas d'objet "element/group" non utilisé :

c'est-á-dire qui ne caractériserait pas un objet "entity" ou un objet "relation" ou un objet "group" car cela serait incohérent par rapport á la définition d'un attribut donnée au chapitre 1.

. Pas d'objet "group" vide :

c'est-á-dire un objet "group" ne possédant aucune relation "CONSIST OF" car cela serait incohérent par rapport á la définition donnée au chapitre 1.

3.3.3 Contraintes portant sur les objets "relation"

. Pas d'objet "relation" qui possède une relation "IDENTIFIED BY" portant sur un objet "role" qui ne correspond pas á un objet "entity" qui participe á cet objet "relation".

. La spécification d'un objet "relation" ne peut contenir une relation "CONSIST OF" portant sur un objet "element/group" inexistant dans le schéma.

3.3.4 Contraintes portant sur les objets "role"

. Présence obligatoire d'un contenu minimal pour les objets "role", constitué des éléments suivants :

  . un nom
  . l'objet "entity" qui joue ce rôle
  . l'objet "relation" au sein duquel le rôle est joué
  . la connectivité associée à l'objet "role" pour l'objet "entity"

. Unicité des noms d'objets "role" (homonymie) : (1)

  deux objets "role" différents ne peuvent porter le même nom.

. Pas d'objet "role" portant des noms différents pour un même contenu (synonymie) :

  un objet "role" ne peut être présent, dans le schéma e-a, à plusieurs reprises.

(1) Une méthode assez simple pour obtenir l'unicité des noms de rôle consisterait à concaténer le nom du rôle et le nom du type d'entité qui joue ce rôle.
Ce travail pourrait être réalisé par un processeur capable de qualifier les noms de manière automatique. Ce processeur devrait posséder une certaine "intelligence" , par exemple dans le cas où un type d'entité joue plusieurs rôles au sein d'un même type d'association. Malheureusement, nous ne possédons pas ce processeur. De plus si nous envisageons cette solution pour les noms de rôle, pourquoi ne pas l'envisager pour les noms d'attribut?

3.3.5 Contrainte portant sur les objets "aggregate"

. Présence obligatoire d'un contenu minimal pour les objets "aggregate", constitué des éléments suivants :

  . un nom
  . une description de l'objet "aggregate" qui devrait contenir sa définition :
  . ses composants
  . sa fonction, c'est-à-dire identifiant ou participant d'une dépendance fonctionnelle

. Unicité des noms d'objets "aggregate" (homonymie) :

  deux objets "aggregate" différents ne peuvent porter le même nom.

. Pas d'objet "aggregate" portant des noms différents pour un même contenu (synonymie) :

  un objet "aggregate" ne peut être présent, dans le schéma e-a, à plusieurs reprises, en portant des noms différents.

3.3.6 Contraintes portant sur les contraintes d'intégrité

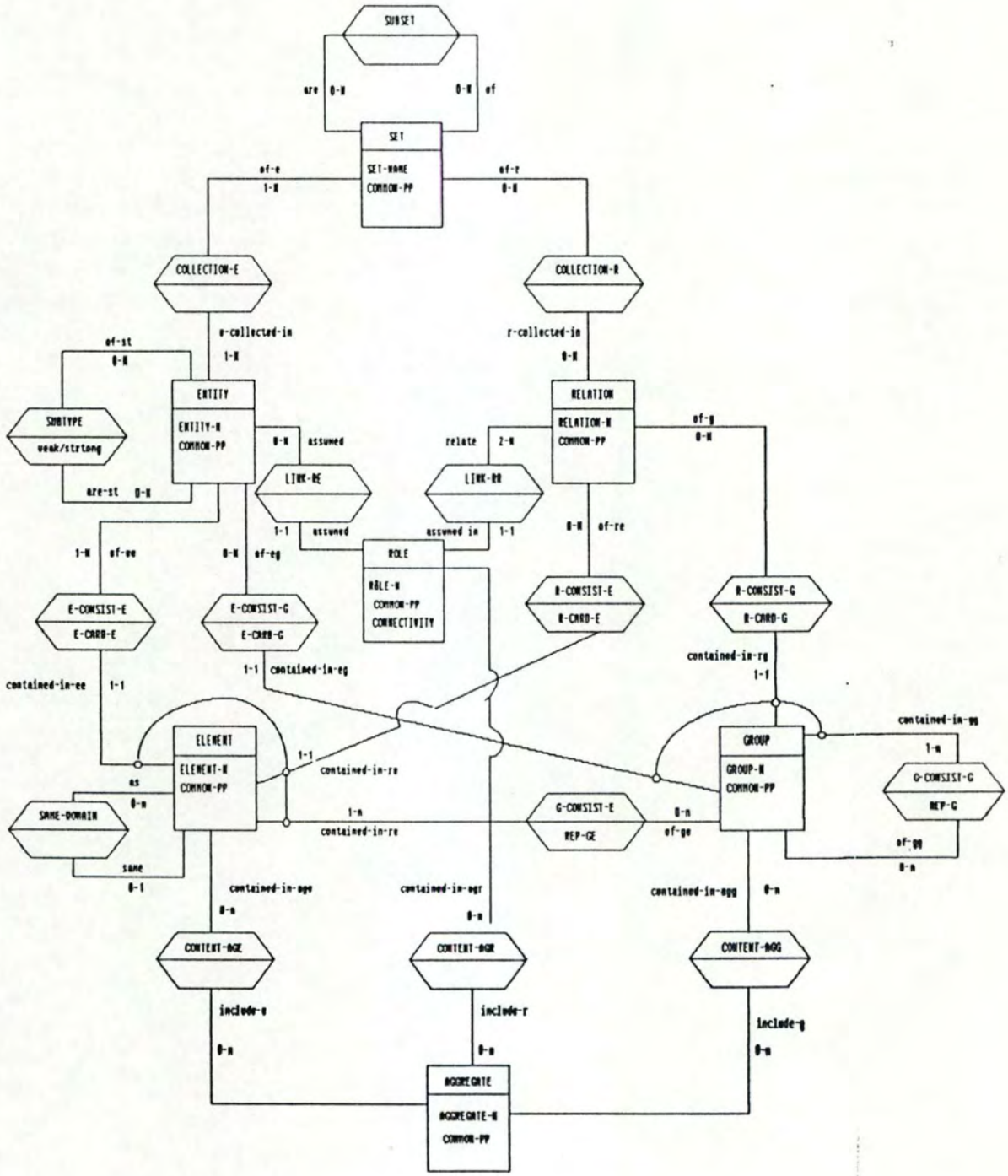
. Le type des hiérarchies de sous-typage d'entité doit être strictement arborescent :

  En effet, si nous permettions les hiérarchies non strictement arborescentes,comme nous l'avons déjà signalé, il serait nécessaire d'introduire la notion de critères de sous-typage. Or, l'état actuel de la théorie dans ce domaine n'est pas assez développée pour permettre un relâchement de cette contrainte.

. Les objets "role" qui apparaissent au sein d'une même dépendance fonctionnelle doivent être joués par des objets "entity" sur le même objet "relation".

3.4 Schéma entité - association du modèle de représentation DSL

3.4.1 Le modèle e-a



Contraintes définitionnelles :

- Tout objet "group" doit au moins participer à 2 occurences d'associations choisies parmis les associations suivantes : "g-consist-g" et "g-consist-e".

- Un AGGREGATE doit au moins participer à une des associations suivantes : . "content-age", "content-agg", "content-agr"

- Tout ENTITY, ELEMENT, GROUP doit posséder . une description qui devrait comprendre : . une définition
  . la durée de vie
  . d'autres caractéristiques décrites au chapître 3 point 1 (niveau complétude)

- Tout AGGREGATE, ROLE, RELATION doit posséder : . une description qui devrait comprendre sa définition
  . d'autres caractéristiques décrites au chapître 3 point 1

Contrainte d'identification :

Toute ENTITY, RELATION doit posséder au moins un ident (cf. identification au point suivant)

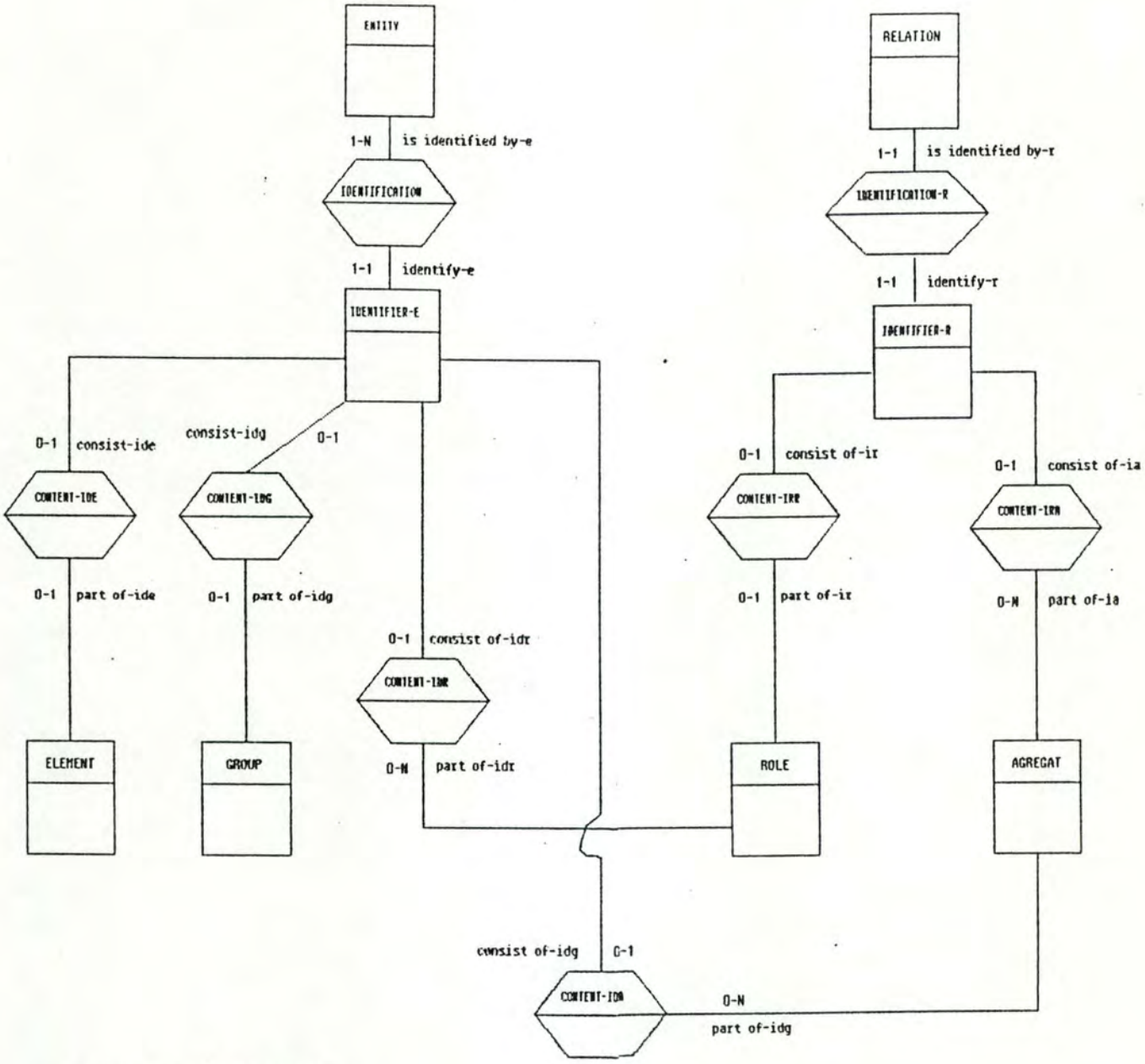
Contrainte additionnelle :

Unicité des noms d'objet, sauf dans le cas ou un même objet "group" ou "element" appartient à différents autres objets "group".

Remarque :

. "common-pp" représente l'ensemble des attributs suivants : description
synonyme
attribute
keywords
see-memo

3.4.2 Le modèle d'identification



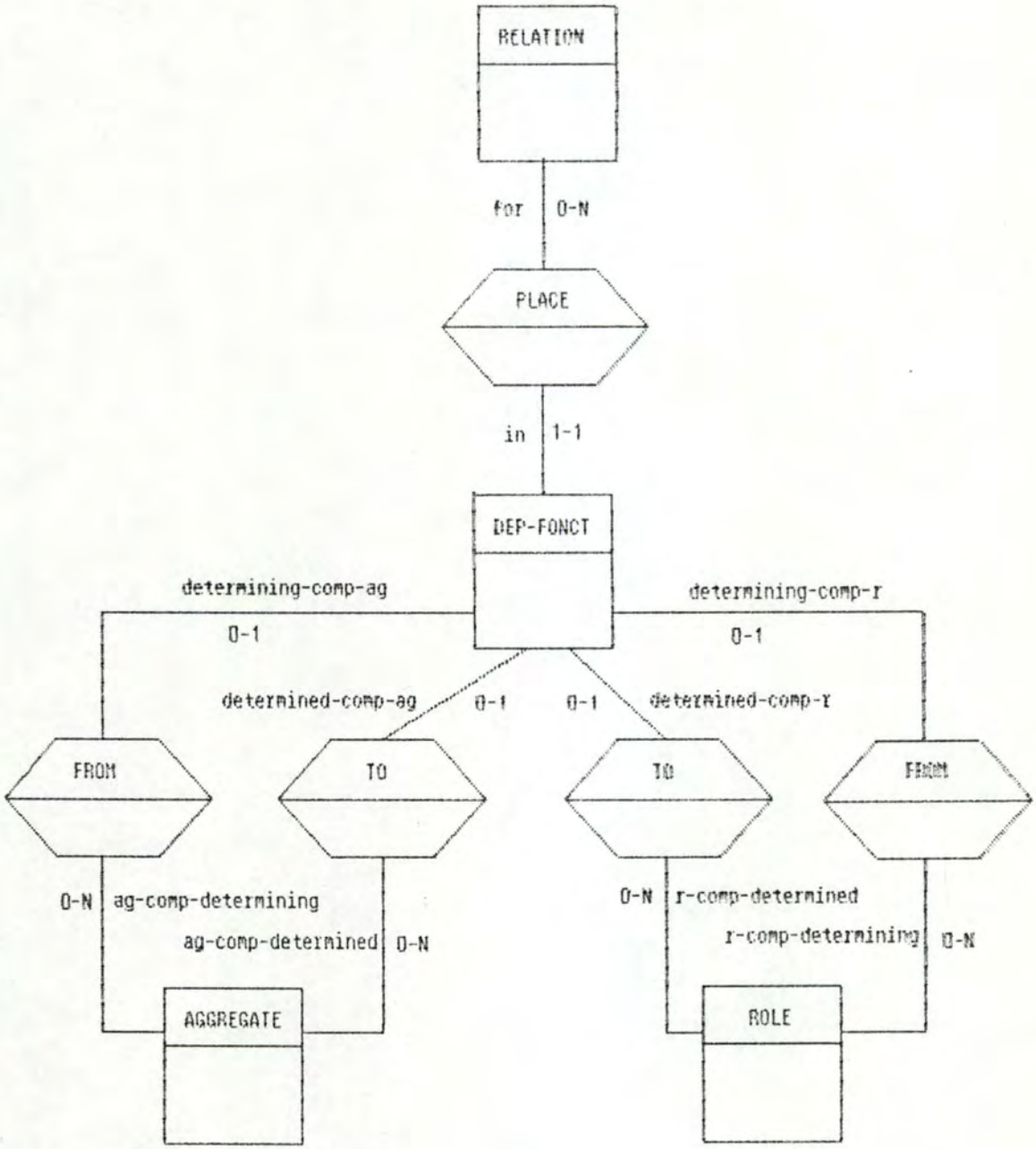
Contraintes additionnelles :

- Un identifiant d'une entité ENTITY doit au moins participer à une des relations suivantes : "content-ide", "content-idg", "content-ida", "content-idr".
- Un identifiant d'une entité RELATION doit au moins participer à une des relations suivantes : "content-irr", "content-idg".
- Un identifiant ne peut être composé que d'entités ELEMENT, GROUP ou ROLE associé à l'entité ENTITY qu'il identifie.

Remarque :

- Si l'on désire créer un identifiant composé de plusieurs objets indépendemment de structure sémantique, il faudra nécessairement créer un AGGREGATE formé de ces objets
- Les entité IDENTIFIER-R et IDENTIFIER-E ne correspondent pas à des objets DSL mais leur présence permet de décrire plus clairement le concept d'identifiant.

3.4.3 Le modèle des dépendances fonctionnelles



Remarque : „Dans la forme canonique, les dépendances fonctionnelles entre objets "element" ou "group" sont interdites, donc seul les dépendances fonctionnelles entre objets "role" au sein d'un objet "relation" sont permises

„Lorsque plusieurs objets "role" déterminent un ou plusieurs autres objets "role", l'ensemble des objets déterminant doit être défini au moyen de l'objet "aggregate"; il en est de même si plusieurs objets "role" constituent le déterminé.

Contraintes : „Un rôle ou un agrégat ne peut composer le déterminé et le déterminant d'une même dépendance fonctionnelle.

DEUXIEME PARTIE :

SYSTEME DE PILOTAGE POUR LA CONCEPTION D'UN SCHEMA CONCEPTUEL D'INFORMATIONS

INTRODUCTION

Cette seconde partie de notre travail sera consacrée à l'étude proprement dite d'un système de pilotage pour la conception d'un schéma conceptuel d'informations.

Ce système sera constitué en grande partie d'une série de tests dont la mise-en-oeuvre permettra de déterminer si le schéma entité - association analysé respecte bien les propriétés de la forme canonique définie au cours de la première partie de notre travail, tests qui seront à la base de l'aide qui sera fournie sous forme de propositions de correction qui devront permettre une meilleure expression de la sémantique du schéma.

En vue d'atteindre cet objectif, nous avons étudié les différentes formes de pilotage et les problèmes qui y sont associés, étude qui vous est présentée au chapitre 4.

C'est à partir de cette analyse que nous avons choisi la forme de pilotage qui nous a paru la plus adéquate par rapport à l'objectif poursuivi.

Dans un second temps, au cours du chapitre 5, nous définirons la série de tests à mettre en oeuvre afin de s'assurer qu'un schéma entité - association, entré dans sa totalité, se trouve bien sous forme canonique. Il s'agit du contrôle à postériori.

Ce n'est qu'ensuite que nous présenterons les différentes approches possibles pour la conception d'un système de pilotage interactif avant d'étudier leur adaptation à l'environnement IDA.

PLAN

DEUXIEME PARTIE : SYSTEME DE PILOTAGE POUR LA CONCEPTION D'UN SCHEMA CONCEPTUEL D'INFORMATIONS

CHAPITRE 4 LES DIFFERENTES FORMES DE PILOTAGE ET LEURS PROBLEMES

4.0 Introduction

4.1 Le système expert

- 4.1.1 Caractéristiques
- 4.1.2 Problèmes

4.2 Le contrôle à postériori

- 4.2.1 Caractéristiques
- 4.2.2 Problèmes

4.3 Le pilotage a posteriori interactif

- 4.3.1 Caractéristiques
- 4.3.2 Problèmes

4.4 Conclusion et choix

CHAPITRE 5 CONTROLE A POSTERIORI : REGLES DE CONTROLE

5.0 Introduction

5.1 tests de complétude

- 5.1.0 Introduction
- 5.1.1 Tests portant sur les objets "entity"
- 5.1.2 Tests portant sur les objets "element"
- 5.1.3 Tests portant sur les objets "group"
- 5.1.4 Tests portant sur les objets "relation"
- 5.1.5 Tests portant sur les objets "role"
- 5.1.6 Tests portant sur les objets "aggregate"

5.2 Tests de contrôle de la redondance

5.2.0 Introduction
5.2.1 Tests portant sur les objets "entity"
5.2.2 Tests portant sur les objets "element"
5.2.3 Tests portant sur les objets "group"
5.2.4 Tests portant sur les objets "relation"
5.2.5 Tests portant sur les contraintes d'intégrité
5.2.6 Tests portant sur les objets "aggregate"

5.3 Règles de cohérence

5.3.0 Introduction
5.3.1 Tests portant sur les objets "entity"
5.3.2 Tests portant sur les objets "element"
5.3.3 Tests portant sur les objets "group"
5.3.4 Tests portant sur les objets "relation"
5.3.5 Tests portant sur les objets "role"
5.3.6 Tests portant sur les objets "aggregate"

5.4 Description d'un scénario de contrôle a posteriori

5.4.0 Introduction
5.4.1 Objectif
5.4.2 Ordonnancement des différents tests

CHAPITRE 6 INTRODUCTION A L'ETUDE DU SYSTEME DE PILOTAGE

CHAPITRE 4 LES DIFFERENTES FORMES DE PILOTAGE ET LEURS PROBLEMES

4.0 Introduction

Ce chapitre est donc consacré à l'étude de différentes formes de pilotage.

En effet, avant de nous diriger vers un type bien précis de pilotage, il nous a semblé plus judicieux d'analyser les différentes approches possibles à notre problème, en corrélation avec les solutions existantes ou en cours de réalisation, étude qui nous permettra de choisir la forme de pilotage qui sera la plus adéquate par rapport au modèle utilisé et à l'objectif poursuivi.

Pour ce faire, nous envisagerons successivement trois approches possibles pour la conception d'un système de pilotage :

- l'approche a priori
- l'approche a posteriori
- l'approche a posteriori interactive

Au cours de cette étude, nous nous axerons principalement sur les caractéristiques et les problèmes liés à ces diverses approches, pour nous permettre d'argumenter le choix de notre propre approche

4.1 L'approche à priori

4.1.1 Caractéristiques

Cette approche consiste à guider l'utilisateur dans le processus d'abstraction lors de la modélisation d'un système d'informations.

Cette guidance peut être envisagée de différentes manières selon le niveau d'abstraction souhaité. Dans un premier temps, nous envisagerons une guidance totale pour un niveau d'abstraction élevé. Ensuite nous passerons à une guidance où l'utilisateur doit déjà avoir réalisé un certain travail d'abstraction par rapport au réel perçu, avant de pouvoir utiliser le système de pilotage.

a) Guidance totale

Cette approche a pour but, à partir d'un énoncé le plus proche possible du langage naturel ou à partir d'un énoncé exprimé à l'aide d'un langage-outil (1), d'aider l'utilisateur à découvrir quels sont les différents concepts qui apparaissent dans l'énoncé (concepts qui sont fonction du modèle choisi).

Le système OICSI [de.23] répond à cet objectif.
A partir d'un texte libre qui décrit le réel perçu, OICSI classe les différentes phrases du texte, après une analyse sémantique et syntaxique, dans trois catégories :

- les phrases objet
- les phrases évènement
- les phrases condition

(Les phrases objet permettent de décrire la structure statique du système d'informations et les deux autres catégories servent à décrire la structure dynamique.)

(1) langage-outil : langage spécialisé parlé dans un domaine spécifique.

Ce classement sera effectué à l'aide de graphes ATN (1).

Une fois le classement réalisé, une première ébauche du schéma est obtenue qui ne tient compte d'aucune forme normale.

Une fois ce schéma général obtenu à l'aide d'un "interpréteur", le "constructeur" (2) réalisera le schéma conceptuel définitif.

D'autres études ont été réalisées dans le but d'établir un lien entre le langage naturel et les concepts du modèle choisi [de.25] .

b) Guidance partielle

Les systèmes basés sur ce type d'approche partent d'un certain nombre d'informations de base extraites du réel perçu, par l'utilisateur et aboutissent au moyen d'interactions avec l'utilisateur à la proposition d'un ou plusieurs schémas. Ce type d'approche demande donc à l'utilisateur un travail plus conséquent que dans la première approche.

De tels systèmes sont également l'objet d'études approfondies, par exemple le système proposé par K. Meyer et J. Doughty [de.05].

Pour ce système, l'information de base est constituée d'une part de spécifications précises d'attributs qui sont vus comme des caractéristiques internes de types d'entité où le type d'entité est une spécification d'une unité de base dans l'organisation analysée.

D'autre part, on spécifiera les rôles existant entre les différents attributs que ce soit des dépendances fonctionnellles, des dépendances multi-valuées, ...

A partir de ces informations, une classification des attributs est opérée pour obtenir finalement des types d'entité, des types d'association et un schéma entité-association.

Un autre exemple est le système expert SECSI [de.22] où l'on part à nouveau des attributs et uniquement, cette fois, des dépendances fonctionnelles pour aboutir à la proposition de plusieurs choix de conception possibles.

---

(1) Graphe ATN : graphe de transition d'états finis qui décrit une séquence type de mots d'une phrase.

(2) Constructeur : Par des transformations successives du réseau sémantique obtenu en sortie de l'interpréteur, le constructeur obtient un schéma relationnel.

4.1.2 Problèmes

Tout le problème dans le cas d'une guidance totale réside dans l'expression en langage naturel de l'énoncé. Ce type d'approche nécessite un outil d'interprétation du langage, avec tous les problèmes engendrés par de tels outils.

Dans certain cas, appel doit être fait à l'utilisateur, soit pour expliciter certains points, soit pour choisir une solution parmis plusieurs. On ne pourra donc parler d'une véritable guidance totale.

Donc ce système, tout comme le système de guidance partielle, ne peut s'appliquer à un utilisateur qui ne connait pas les notions de base relatives à la conception d'un schéma conceptuel.

Dans ce cas, un système dont l'approche est à priori, n'est-il pas un peu lourd et inutile ?

4.2 Le contrôle à postériori

4.2.1 Caractéristiques

Comme le sous-entend son nom, ce type d'approche consiste à réaliser une vérification sur un schéma conceptuel déjà créé et non en cours de construction. Les résultats de cette validation sont fournis à l'utilisateur afin de mettre à profit ces conclusions pour corriger le schéma analysé.

Cette méthodologie que l'on retrouve par exemple dans le système EASY ER de Fabrizio M. Ferrara [de.14] reflète le processus de conception généralement utilisé qui consiste à modéliser le système d'informations à partir des besoins spécifiés par les utilisateurs, pour ensuite soumettre la solution obtenue à l'approbation de ces utilisateurs.

A nouveau, nous avons pu constaté que les erreurs signalées, et les corrections possibles proposées portent essentiellement sur les violations des contraintes intrinsèques(1) au modèle utilisé et non pas sur un ensemble de propriétés qui permettent l'amélioration de l'expression de la sémantique contenue dans le schéma analysé.

4.2.2 Problèmes

Si cette méthodologie reflète bien la pratique souvent utilisée, elle s'avère peu adaptée lorsque l'utilisateur d'un tel système ne maîtrise pas parfaitement les règles de construction à respecter.

En effet, dans ce cas, le contrôle à postériori aura pour conséquence la détection d'un nombre important d'erreurs, ce qui demandera des modifications tout aussi importantes à apporter au schéma analysé et un nombre important de réexécutions du programme de validition.

Par contre, la maîtrise de ces règles constitue un avantage considérable dans ce type d'approche, dans la mesure où les tests de validation, souvent inutiles dans ce cadre, ne viennent pas ralentir le processus de construction du schéma, et d'autre part, permetteront la détection d'erreurs qui ne demanderont que quelques "retouches" bénignes au schéma.

---

(1) Par contraintes intrinsèques, nous entendons les contraintes de complétude, de redondance et de cohérence liées au modèle.

4.3 Le pilotage à posteriori interactif

4.3.1 Caractéristiques

Comme dans la première approche que nous avons étudiée, le pilotage interactif possède des outils de création et/ou de modification d'un schéma conceptuel, et comme dans l'approche précedente, il possède également des outils de validation d'un schéma par rapport à certains critères.

Ce qui différencie cette approche de la précédente, c'est d'une part, le fait que les tests de validation sont effectués durant la phase de construction du schéma conceptuel, et dès qu'une erreur est détectée, elle est signalée à l'utilisateur avec ses origines et corrections possibles. Et d'autre part, les corrections sont effectuées soit au moyen d'un apport d'informations de l'utilisateur, soit entièrement par ce dernier, dès l'apparition du problème.

Cette approche se distingue de la première étudiée par le fait que le travail d'abstraction doit être réalisé par l'utilisateur à partir du réel perçu. Le système n'a pour but que de vérifier l'exactitude des informations entrées, par rapport à des règles de construction d'un schéma et de proposer des solutions en cas d'erreur. Si l'on reprend le système OICSI, l'approche à posteriori interactive se limite plus ou moins à la dernière phase, réalisée par le "constructeur" de OICSI [de.23].

A nouveau, il est intéressant de signaler que parmi les systèmes étudiés [de.18, de.19], les règles à respecter sont essentiellement des contraintes intrinsèques au modèle utilisé.

4.3.2 Problèmes

Inversement à l'approche "contrôle à postériori", ce type d'outil sera beaucoup plus appréciable pour un utilisateur qui ne maîtrise pas les critères à respecter car il constitue un guide efficace qui signale les erreurs au moment opportun afin d'éviter des corrections trop importantes au schéma conceptuel, et d'autre part, il permet l'acquisition progressive des connaissances de base nécessaires pour la modélisation.

Par contre, bon nombre des tests effectués durant la construction du schéma conceptuel s'avèreront superflus pour un utilisateur "expert" dans les règles à respecter.

4.4 Conclusion et choix

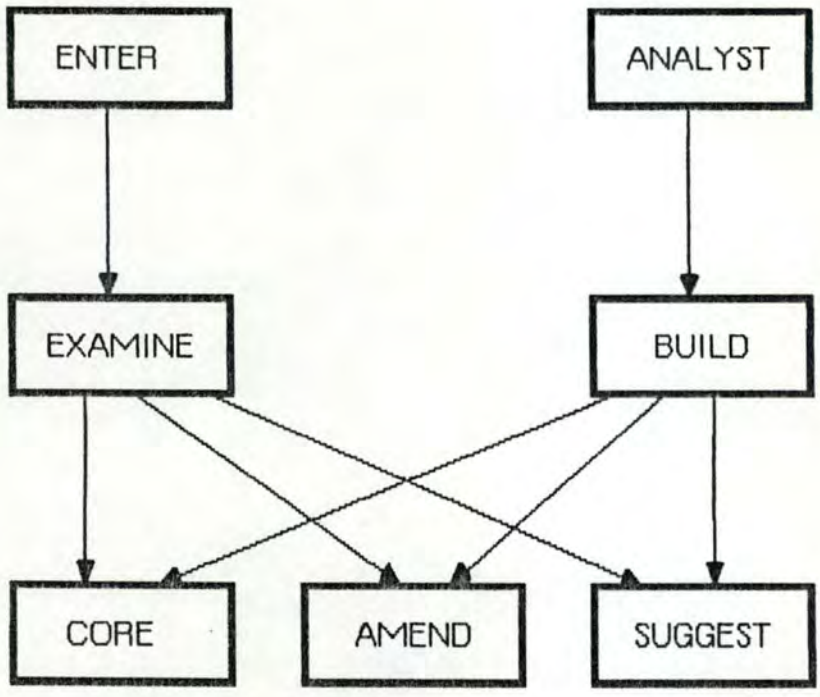
Le choix d'un système de pilotage qui suit l'approche à priori, nous semble inapproprié car notre système s'adressera non pas à des néophites mais à des personnes qualifiées. Dès lors, il nous semble inutile de faire appel à un système aussi complexe et imprécis quant au résultat obtenu (1) .

C'est pourquoi, nous nous sommes orientés vers un autre type de pilotage.

Dans le but d'éviter les désavantages des approches "contrôle à postériori" et "pilotage interactif", tout en bénéficiant de leurs avantages, il nous a semblé fort opportun d'opter pour la conception d'un système de pilotage qui associe ces deux approches.

Et donc, le système de pilotage que nous nous proposons de réaliser sera basé sur la possibilité de mettre en oeuvre ces différents scénari, de la même manière que le système de I. T. Hawryszkiewycz [de.13].

Le système d'aide à la conception d'un schéma conceptuel e-a proposé par cet auteur a l'intérêt de présenter deux approches possibles , d'une part l'approche à posteriori et d'autre part, l'approche à posteriori interactive, approches qui peuvent être perçues facilement sur le schéma représentant la structure de ce système :



(1) cf. le classement des phrases dans OICSI (pg. 2.18 §2.3.2 [de.23]).

- Dans le système à posteriori, l'utilisateur entre la spécification de son schéma au moyen du module "ENTER". Ce n'est qu'ensuite que le module "EXAMINE" permettra d'effectuer la validation du schéma.

  Cette validation se fait sur base d'un certain nombre de propriétés que doit posséder tout schéma e-a, l'ensemble de ces propriétés forment les caractéristiques de la deuxième forme normale (1).

- Dans l'approche à posteriori interactive, l'utilisateur entre en dialogue avec le système par l'intermédiaire d'un interface en langage naturel où "ANALYST" interroge l'utilisateur en termes de vues du réel perçu plutôt qu'en termes de concepts entité-association et interprète ensuite la réponse en termes de concepts e-a.
  La guidance est basée sur un certain nombre d'interprétations sémantiques des diverses violations qui sont utilisées pour entrer en communication avec l'utilisateur et proposer des suggestions de modifications.

NB : CORE est le module de recherche des violations.

SUGGEST contient un ensemble de questions suggérant les changements possibles du schéma.

AMEND contient le schéma éventuellement corrigé.

BUILD enregistre le schéma avec validation directe au moyen de CORE .

Pour réaliser notre outil d'aide, nous nous proposons d'étudier plus en détail l'approche "contrôle à postériori", dans une première phase, afin de déterminer la série de tests à mettre en oeuvre pour s'assurer de la validité d'un schéma entité - association par rapport à la forme canonique. Et dans une seconde phase, nous introduirons les différentes approches possibles de pilotage a posteriori interactif.

(1) Cf. Préambule du chapitre 2

CHAPITRE 5 CONTROLE A POSTERIORI : REGLES DE CONTROLE

5.0 Introduction

Maintenant que nous nous sommes fixés sur l'approche que nous allons suivre pour développer le système de pilotage, nous allons dans un premier temps étudier la première partie de cette approche, c'est-à-dire le contrôle à posteriori, qui nous servira de base pour l'analyse de la seconde partie du système.

Au cours d'une première phase, nous dériverons de chaque propriété définie dans la forme canonique aux points 2.1, 2.2, 2.3 et des restrictions à l'utilisation de DSL définie au point 3.2.2, l'ensemble des tests (1) qui permettront de vérifier qu'un schéma e-a possède bien cette dernière. De plus, à chaque test obtenu, nous associerons le(s) interprétation(s) sémantique(s) possible(s) liée(s) au non-respect de cette propriété, ainsi que les corrections éventuelles à effectuer. L'ensemble des tests sera structuré en fonction des différents critères déjà utilisés lors de la description de la forme canonique, c'est-à-dire la complétude, la redondance et la cohérence.

Au cours d'une seconde phase, nous proposerons un regroupement possible des différents tests obtenus lors de la première phase, ce regroupement ayant pour objectif la réduction du nombre d'accès à la base de données IDA qui seront nécessaires pour la validation d'un schéma e-a.

---

(1) Tous les tests spécifiés dans ce chapitre sont des tests généraux définis pour un outil de "design", ils sont indépendants de tout outil d'interface avec une base de données tel celui realisé par Monsieur Hainaut.
Ces tests peuvent être complétés par ceux exprimés dans l'annexe D au point 3, si l'on désire utiliser un outil de design utilisant l'extracteur spécifié dans l'annexe D au point.1.

5.1 Les tests de complétude

5.1.0 Introduction

A partir des règles définies au point 2.1, nous allons définir les tests à réaliser pour vérifier que ces différentes règles sont respectéees par le schéma e-a traité.

5.1.1 Tests portant sur les objets "entity"

Pour tout objet "entity", les tests suivants devront être réalisés :

tst 1-c : Sélection des objets "entity" pour lesquels ..... il n'existe pas de relation DSL "DESCRIPTION; ..."

Diagnostic :

La spécification des objets sélectionnés est incomplète, il manque la définition constitutive et la durée de vie de ces objets.

Correction :

Pour chaque objet sélectionné, on doit déclarer une clause "DESCRIPTION; ..." qui reprend la définition et la durée de vie de l'objet.

Remarque :

La durée de vie et la définition d'un objet doivent être spécifiées au sein de la relation "DESCRIPTION; ..." à l'aide d'un texte libre.
A l'heure actuelle, nous ne disposons pas de moyens d'analyse sémantique qui nous permettent de valider un texte.
Nous pourrons donc valider la présence de la relation "DESCRIPTION; ..." mais pas son contenu.

tst 2-c : Sélection des objets "entity" qui ne possèdent pas de relation " IDENTIFY BY ...".

Diagnostic :

La spécification des objets sélectionnés est incomplète, aucun identifiant n'est défini pour ces objets.

Correction :

Pour chaque objet sélectionné, on doit déclarer au moins une clause "IDENTIFY BY ..." où est spécifié un identifiant correct pour l'objet "entity" sélectionné.

tst 3-c : Sélection des objets "entity" qui ne possèdent pas de relation "CONSIST OF ...".

Diagnostic :

La spécification des objets sélectionnés est incomplète, aucun objet "element" ou "group" n'est défini comme attribut de ces objets.

Correction :

Pour chaque objet sélectionné, on doit déclarer au moins une clause "CONSIST OF ..." où est spécifié au moins un attribut de l'objet "entity" sélectionné.

5.1.2 Tests portant sur les objets "element"

Pour tout objet "element", les tests suivants devront être réalisés :

tst 4-c : Sélection des objets "element" pour lesquels ..... il n'existe pas de relation DSL "DESCRIPTION; ..."

Diagnostic :
idem (tst 1-c) en termes d'objet "element".

Correction :
idem (tst 2-c) en termes d'objet "element".

Remarque :
idem (tst 1-c) en termes d'objet "element"

tst 5-c : Sélection des objets "element" pour lesquels ..... ni une relation "DOMAIN OF VALUE ...", ni une relation "SAME DOMAIN AS ..." n'est définie.

Diagnostic :
La spécification des objets sélectionnés est incomplète, aucun domaine de valeurs n'est défini pour ces objets.

Correction :
Pour chaque objet "element" sélectionné, il faut spécifier un domaine de valeurs au moyen d'une clause "DOMAIN OF VALUE ..." où "SAME DOMAIN AS ...".

tst 6-c : Sélection des objets "element" pour lesquels
..... il n'existe pas de relation "CONTAINED IN...".

Diagnostic :

La spécification des objets sélectionnés est incomplète, car ces objets tels qu'ils sont spécifiés, ne sont la propriété d'aucun objet "entity", "relation" ou "group".

Correction :

Pour chaque objet "element" sélectionné, il faut spécifier l'objet "entity", "relation" ou "group" que cet objet "element" détermine.

tst 7-c : Sélection des objets "element" pour lesquels
..... il existe une relation "CONTAINED {sys-par} TIMES IN ..." mais pour laquelle {string }

la valeur de {sys-par} ne corespond pas au moins à une des exigences suivantes :

- le nom du système paramètre correspond à la structure suivante:

  "sp-<min>-<max>-[<identificateur>]"

- le système-paramètre comprend l'expression d'un intervalle dans une clause "RANGE VALUE IS..." .

Diagnostic :

La répétitivité et le caractère obligatoire ou facultatif des objets sélectionnés sont mal spécifiés.

Correction :

Les valeurs données à { sys-par} ou {string} dans la clause :

"CONTAINED {sys-par} TIMES IN ...",
{string }

doivent être corrigées de manière à obéir à ces exigences.

Remarque :

Ces règles correspondent à une expression minimale obligatoire, d'autres éléments peuvent être spécifiés via cette relation.

ex : La probabilité d'avoir une valeur nulle

5.1.3 Tests portant sur les objets "group"

Pour tout objet "group", les tests suivants devront être réalisés :

tst 8-c : idem (tst 4-c) mais en termes d'objets "group".

tst 9-c : idem (tst 6-c) mais en termes d'objets "group".

tst 10-c : idem (tst 7-c) mais en termes d'objets "group".

tst 11-c : Sélection des objets "group" pour lesquels une ou plusieurs relations "CONSIST OF" qui portent, en tout, sur une combinaison d'au moins deux objets "element" et/ou "group" n'existent pas.

Diagnostic :

La spécification des objets "group" sélectionnés est incomplète, car ces objets "group" ne sont pas composés d'au moins deux objets "element" et/ou "group".

Correction :

Une ou plusieurs relations "CONSIST OF ..." doivent être spécifiées afin de compléter la spécification des objets "group" sélectionnés par les objets "element" ou "group" sur lesquels ils portent.

5.1.4 Tests portant sur les objets "relation"

Pour tout objet "relation", les tests suivants devront être réalisés :

tst 12-c : idem (tst 1-c) mais en termes d'objets "relation".

tst 13-c : Sélection des objets "relation" qui ne possèdent pas au moins deux relations "RELATES ..." telles que celle-ci :

"RELATES entity-name AS role-name [WITH CONNECTIVITY { sys-par }]
{ string }

Diagnostic :

La spécification des objets "relation" sélectionnés est incomplète car ces objets ne portent pas sur au moins deux objets "role", rôles joués par un ou deux objets "entity".

Correction :

Pour chaque objet "relation" sélectionné, il faut spécifier une ou plusieurs clauses "RELATES ..." de manière à compléter la spécification de ces objets "relation".

5.1.5 Tests portant sur les objets "role"

Pour tout objet "role", les tests suivants devront être réalisés :

tst 14-c : Sélection des objets "role" pour lesquels il n'existe pas de relation "ASSUMED BY entity-name IN relation-name WITH CONNECTIVITY ...".

Diagnostic :

La spécification des objets sélectionnés est incomplète, il faut spécifier l'objet "entity" qui joue le rôle, et l'objet "relation" au sein duquel le rôle est joué et avec quelle connectivité.

Correction :

Il faut spécifier pour chaque objet "role" sélectionné une clause "ASSUMED BY ..." ou bien compléter la clause qui existe déjà.

5.1.6 Tests portant sur les objets "aggregate"

Pour tout objet "aggregate", les tests suivants devront être réalisés :

tst 15-c : Sélection des objets "aggregate" pour lesquels il n'existe pas de relation "CONSIST OF ..." qui portent au moins sur deux objets.

Diagnostic :

La création d'un objet "aggregate" n'a pas de sens s'il ne sert pas à regrouper plusieurs objets.

Correction :

Il faut spécifier, pour chaque objet sélectionné, les objets qui composent cet objet "aggregate".

tst 16-c : Sélection des objets "aggregate" pour lesquels il n'existe pas de relation "IDENTIFY ...", "IDENTIFIED BY ...", "FUNCTIONNALY DEPENDENT ON ..." ou "FUNCTIONNALY DETERMINES ...".

Diagnostic :

La création d'un objet "aggregate" n'a pas de sens s'il n'est pas utilisé.

Correction :

Il faut pour chaque objet sélectionné spécifier une ou plusieurs clauses "IDENTIFY ...", "FUNCTIONNALY ...".

5.2 Tests de contrôle de la redondance

5.2.0 Introduction

A partir des règles de contrôle de la redondance définies aux points 2.2 et 3.3, nous allons définir, dans ce point, les tests à réaliser pour vérifier qu'il n'y a pas de redondance dans le schéma traité.

5.2.1 Tests portant sur les objets "entity"

Pour tout objet "entity", les tests suivants devront être réalisés :

tst 1-r : Sélection pour chaque objet "entity" des identifiants qui ne sont pas minimaux, c'est-à-dire que l'on va sélectionner les identifiants au sein desquels il existe des dépendances fonctionnelles.
L'identifiant d'un objet "entity" est composé soit d'un objet "element" ou "group", soit d'un objet "aggregate" lui-même composé d'objets "element", "group" et/ou "role". Dans le cas où l'identifiant est un objet "aggregate", on vérifiera que la relation "CONSIST OF ..." de l'objet "aggregate" ne porte pas sur des objets pour lesquels, il existe une ou plusieurs relations "FUNCTIONNALY DEPENDENT ON ..." qui portent également sur des objets qui composent l'identifiant de l'objet "entity".

Diagnostic :

Les objets sélectionnés pour chaque objet "entity", ne sont pas des identifiants minimaux.

Correction :

Les relations "IDENTIFIED BY ..." sélectionnées doivent être supprimées ou corrigées de manière à obtenir l'identifiant minimal de l'objet "entity".

tst 2-r : Sélection de tout objet "entity" pour lequel il existe plusieurs relations "IDENTIFIED BY ..." qui portent sur des objets "aggregate" dont la composition ("CONSIST OF ....") d'un de ces objets est incluse dans la composition d'un autre.

Diagnostic :

Il existe pour les objets sélectionnés, des identifiants qui ne sont pas minimaux, car il existe une dépendance fonctionnelle entre les objets qui composent ces identifiants.

Correction :

Les relations "IDENTIFIED BY ..." incorrectes pour les objets sélectionnés, doivent être supprimées ou corrigées de manière à obtenir l'identifiant minimal de l'objet "entity". (1)

(1) cf. Annexe A.4

5.2.2 Tests portant sur les objets "element"

Pour tout objet "element" les tests suivants devront être réalisés :

tst 3-r : Sélection pour chaque objet "element" des autres objets "element" ayant même contenu minimal c'est-à-dire sélection des objets "element" pour lesquels les relations "DOMAIN OF VALUE ..." ou "SAME DOMAIN AS ...", "CONTAINED ..." et "ATTRIBUTE ..." ou "KEYWORD ..."(pour la dérivabilité) sont identiques.

Diagnostic :

Si pour un objet "element" la liste des objets sélectionnés n'est pas vide, il se peut que les objets qui la composent soient redondants (c'est-à-dire : synonyme).
Seul le contenu de la relation "description" permet de déterminer s'ils le sont ou s'ils ne le sont pas (1).

Correction :

Dans le cas des synonymes, tous les objets "element" redondants doivent être supprimés.

(1) Il faut également tenir compte ici, de la remarque faite pour le test (tst 1-c)

5.2.3 Tests portant sur les objets "group"

tst 4-r : Pour tout objet "group", le test (tst 3-r) devra être réalisé. Une vérification supplémentaire s'impose : il faut comparer les relations "CONSIST OF ...", alors que la comparaison des relations "DOMAIN OF VALUE ..." ou " SAME DOMAIN AS ..." et la comparaison au niveau de la dérivabilité sont exclues.

Diagnostic :
idem (tst 2-r) mais en termes d'objet "group".

Correction :
idem (tst 2-r) mais en termes d'objet "group".

5.2.4 Tests portant sur les objets "relation"

Pour tout objet "relation", les tests suivants devront être réalisés :

tst 5-r : Sélection des objets "relation" de degré qui pourrait être redondant avec une composition d'objets "relation" qui portent sur les mêmes objets "entity" que ceux sur lesquels porte l'objet "relation" sélectionné.

Diagnostic :

Les objets "relation" sélectionnés peuvent être redondants par rapport à la composition d'autres objets "relation". Ceci ne peut être décidé qu'à partir de la description de ces objets.

Correction :

Les objets "relation" redondants doivent être supprimés.

---

(1) Si les objets "entity" A et B sont reliés par un objet "relation" a et si les objets "entity" B et C sont reliés par un objet "relation" b alors on dira qu'il existe une composition de deux objets "relation" qui relie A et C.
Si A et X sont relié par n-1 objets "relation" et s'il existe un objet "relation" qui relie X et Y, alors on dira qu'il existe une composition de n objets "relation" entre A et Y.

---

tst 6-r : Sélection pour chaque objet "relation" des
..... identifiants qui ne sont pas minimaux c'est-à-dire que l'on va sélectionner les identifiants au sein desquels il existe des dépendances fonctionnelles.
L'identifiant d'un objet "relation" est composé soit d'un objet "role", soit d'un objet "aggregate" lui-même composé d'objets "role". Dans le cas d'un objet "aggregate on vérifiera que la relation "CONSIST OF" qui porte sur l'objet "aggregate" ne porte pas également sur des objets "role" pour lesquels il existe une ou plusieurs relations "FUNCTIONNALLY DEPENDENT ON ..." qui portent également sur des objets qui composent l'identifiant de l'objet"relation".

Diagnostic :

Les objets sélectionnés pour chaque objet "relation" ne sont pas des identifiants minimaux.

Correction :

Les relations "IDENTIFIED BY ..." sélectionnées doivent être supprimées ou corrigées de manière à obtenir l'identifiant minimal de la relation.

5.2.5 Tests portant sur les contraintes d'intégrité

tst 7-r : Sélection des contraintes d'intégrité redondantes (1,2) .

tst 8-r : Sélection des dépendances fonctionnelles transitives.

(1) La sélection des contraintes d'intégrité incohérentes pourrait se faire grâce à un moteur d'inférence réalisé par M. Lorant [Contributions à un système de dialogue homme-machine à composantes orales, chapitre 5]

(2) Lors de notre stage à Lyon, dans l'approche que nous avons choisi, nous avons préalablement étudié les cas d'incohérence et de redondance possible avant de spécifier les différents tests à effectuer pour détecter ces erreurs, ainsi que les moments de déclenchement de ces test (cf.annexe E).
L'approche de M. Lorant diffère de notre travail, dans la mesure où après quelques définitions des concepts de base apparaissant dans les contraintes en termes de domaine de valeurs, il met en oeuvre un moteur d'inférence utilisant le principe de démonstration par réfutation pour détecter les incohérences et redondances.

5.2.6 Test portant sur les objets "aggregate"

Pour tout objet "aggregate", le test suivant devra être réalisé :

tst 9-r : Sélection, pour chaque objet "aggregate" des objets "aggregate" ayant le même contenu, c'est-à-dire la même relation "CONSIST OF ....".

Diagnostic :

Toute liste non vide associée à un objet "aggregate" est composée d'objets synonymes.

Correction :

Il faut compléter un objet par les relations "FUNCTIONNALY ..." ou "IDENTIFY ..." des autres objets repris dans la liste et en suite on supprime les objets de cette liste.

5.3 Tests de contrôle de la cohérence

5.3.0 Introduction

A partir des règles définies aux points 2.3 et 3.3, nous allons définir, dans ce point, les tests à réaliser pour vérifier que le schéma est cohérent.

5.3.1 Tests portant sur les objets "entity"

Pour tout objet "entity", les tests suivants devront être réalisés :

tst 1-h : Sélection des objets "entity" pour lesquels il existe une relation "IDENTIFIED BY ..." qui porte :

a) sur un objet "element" ou "group" dont la relation "CONTAINED IN ..." ne porte pas sur l'objet "entity" traité.

ou

b) sur un objet "role" dont la relation "ASSUMED BY ... IN {relation-name} ..."ne porte pas sur un objet "relation" pour lequel il existe une relation "RELATES {entity-name} ..." portant sur l'objet "entity" traité.

ou

c) sur un objet "aggregate" dont la relation "CONSIST OF ..." porte :

1) cf. le point a) précédent mais pour un ou plusieurs objets "group" et/ou "element".

ou

2) cf. le point b) précédent mais pour un ou plusieurs objets "role".

Diagnostic :

L'identifiant porte sur des objets qui n'appartiennent pas à l'objet "entity" qu'il identifie ou ces objets n'ont pas été déclarés comme propriété de celui-ci.

Correction :

Selon l'erreur qui est détectée soit l'identifiant des objets sélectionné doit être modifié, soit les objets sur lesquels ils portent doivent être corrigés.

test 2-h : Sélection de tout objet "entity" pour lequel ..... il existe une intersection non vide entre l'ensemble des objets "entity" qui participent directement ou par transitivité à une relation "SUBTYPE OF ... " de cet objet et l'ensemble des objets "entity" qui participent directement ou par transitivité à la relation "SUBTYPE ARE ..." de cet objet (1).

Diagnostic :

Les objets sélectionnés sont à la fois fils et père d'un autre objet "entity" dans la hiérarchie des sous-types.

Correction :

La hiérarchie de sous-type à laquelle l'objet "entity" sélectionné appartient doit être corrigée de manière à obtenir une hiérarchie sans cycle.

(1) La relation de sous-typage possède la propriété de transitivité: un sous-type d'un type d'entité A qui est lui même sous-type d'un type d'entité B est également sous-type du type d'entité B.

tst 3-h : Sélection, pour chaque objet "entity", des
..... objets "element" ou "group" sur lesquels porte une relation "CONSIST OF ...", si ceux-ci ne sont pas définis dans la base de données.

Diagnostic :

Les objets sélectionnés pour chaque objet "entity" ne sont pas spécifiés ou ils sont erronés.

Correction :

Spécification des objets sélectionnés ou suppression de ceux-ci dans la relation "CONSIST OF ..." de l'objet "entity".

Remarque :

En DSL, les objets de type "UNDEFINE" sont automatiquement traités.
De ce fait, ce test devient inutile.

5.3.2 Tests portant sur les objets "element"

Pour tout objet "element", les tests suivants devront être réalisés :

tst 4-h : Sélection des objets "element" qui participent à une relation "FUNCTIONNALY DEPENDENT ON ..." en tant que "déterminé" et pour lesquels, le "déterminant" de la dépendance fonctionnelle n'identifie pas l'objet auquel l'objet "element" traité appartient.
Parmi les objets sélectionnés, on distingue
- ceux dont le "déterminant " de la relation "FUNCTIONNALY DEPENDENT ON ..." (ou un des objets qui le composent) appartient à un objet "aggregate" qui dentifie l'objet auquel appartient l'objet sélectionné
- ceux dont :
  soit le "déterminant" de la relation "FUNCTIONNALY DEPENDENT ..." fait partie des objets sur lesquels porte une relation "CONSIST OF ..." mais pas "IDENTIFY ..." de l'objet auquel appartient l'objet sélectionné
  soit le "déterminant de la relation "FUNCTIONNALY DEPENDENT ..." est un objet "role" joué au sein de l'objet "relation" qui porte sur l'objet auquel appartient l'objet sélectionné,
- ceux dont le "déterminant" ne correspond à aucun des cas précédents.

Diagnostic :
Dans le premier cas, les objets sélectionnés ne dépendent pas entièrement de l'identifiant de l'objet auquel appartient l'objet "element".
Il y a sans doute aggrégation de deux concepts distincts.
Dans le deuxième cas, les objets sélectionnés dépendent fonction--nellement d'un objet non identifiant pour les objets auxquels ils appartiennent, ce qui est interdit.
Dans le troisième cas, les objets sélectionnés dependent fonctionnellement d'un objet qui n'est pas associé aux objets auxquels ils appartiennent.

Correction :

Dans le premier cas, il faut décomposer l'objet "entity" auquel appartient l'objet "element" sélectionné selon les règles données dans l'annexe A.1.
Dans le second et le troisième cas, il faut détruire les dépendances fonctionnelles interdites.

tst 5-h : Sélection des objets "element" pour lesquels
..... dans la relation
"CONTAINED {sys-par} TIMES IN ...",
{string }
si une valeur est donnée au "sys-par" ou au "string", la valeur maximale rencontrée dans ceux-ci est supérieure à 1.

Diagnostic :

Les éléments sélectionnés sont répétitifs, cela est souvent le signe de l'aggrégation de deux concepts distincts.

Correction :

Il faut décomposer l'objet qui apparaît dans la clause "CONTAINED IN ..." de l'objet "element" sélectionné, selon les règles enoncées dans l'annexe A.2.

tst 6-h : Sélection des objets "element" sur lesquels
..... portent plusieurs relations "CONTAINED IN ..." relatives à des objets qui ne sont pas tous des objets "group".

Diagnostic :

Etant donné l'unicité des noms, un objet "element" ne peut appartenir à plusieurs objets distincts excepté si ce sont tous des objets "group".
Les objets sélectionnés ne respectent pas cette règle.

Correction :

Les différents objets "element" sélectionnés doivent porter des noms distincts selon qu'ils appartiennent à tel ou tel objet.

5.3.3 Tests portant sur les objets "group"

Les tests décrits au point 5.3.2 devront , également, être réalisés pour tout objet "group".

5.3.4 Tests portant sur les objets "relation"

Pour tout objet "relation" , les tests suivants devront être réalisés.

tst 7-h : Sélection des objets "relation" pour lesquels la relation "IDENTIFY ..." porte sur un objet "role" ou sur un objet "aggregate" formé d'objets "role" dont les relations "ASSUMED BY ..." ne portent pas sur l'objet "relation" sélectionné.

Diagnostic :

Un objet "relation" ne peut être identifié que par des objets "role" joués en son sein.

Correction :

L'identifiant des objets sélectionnés doit être modifié.

tst 8-h : Sélection des objets "relation" pour lesquels la relation "IDENTIFIED BY ... " ne porte pas sur un objet "role" ou sur un objet "aggregate" dont la relation "CONSIST OF ..." ne porte pas uniquement sur des objets "role".

Diagnostic :

Les objets sélectionnés possèdent des identifiants qui ne sont pas composés uniquement d'objets "role". Ces identifiants sont donc incorrects.

Correction :

Au moins un des identifiants des objets sélectionnés doit être corrigé soit en supprimant tout l'identifiant soit en supprimant les objets qui ne sont pas des objets "role".

tst 9-h : Sélection des objets "relation" pour lesquels la cardinalité contenue dans la relation "CONTAINED {sys-par} IN ...} ne répond pas à la règle énoncée précédemment. Ce test ne peut être effectué que si toutes les données nécessaires sont présentes.

Diagnostic :

La cardinalité des objets sélectionnés est en contradiction avec la cardinalité des objets "entity" et la connectivité des objets "role" sur lesquels ils portent.

Correction :

La cardinalité des objets sélectionnés doit être respécifiée.

tst 10-h : Sélection des objets qui répondent aux règles de décomposition énoncées dans l'annexe A.1.

Diagnostic :

Les objets "relation" sélectionnés ne possèdent pas un degré minimun.

Correction :

Les objets sélectionnés doivent être décomposés selon les règles énoncées dans l'annexe A.1.

5.3.5 Tests portant sur les objets "role"

Pour chaque objet "role" les tests suivants devront être réalisés :

tst 11-h : Sélection des objets "role" sur lesquels porte une relation "IDENTIFY ... " et dont la connectivité qui apparaît dans la relation "ASSUMED BY ... IN ... WITH CONNECTIVITY ..." est différente de " 0-1" ou "1-1"
ou
dont l'objet "relation" qui apparaît dans cette même relation porte sur plus de 2 relations "RELATES ...".

Diagnostic :
La composition de l'identifiant des objets sélectionnés est incorrecte, elle ne correspond pas à la définition d'un identifiant et aux restrictions qui y sont apportées.

Correction :
Il faut corriger les identifiants incorrects pour les objets sélectionnés.

tst 12-h : Sélection des objets "role" pour lesquels il existe une relation "FUNCTIONNALY DEPENDENT ON ..." dont les différents objets "role" intervenant ne sont pas joués au sein du même objet "relation" c'est-à-dire que les objets sur lesquels porte leur relation "ASSUMED BY ... IN ..." sont différents.

Diagnostic :
Les dépendances fonctionnelles spécifiées au sein des objets sélectionnés sont incorrectes.

Correction :
Correction des dépendances fonctionnelles incorrectes.

5.3.6 Tests portant sur les objets "aggregate"

Pour chaque objet "aggregate", les tests suivants devront être réalisés :

tst 13-h : Sélection des objets "aggregate" composés uniquement d'objets "role" pour lesquels il existe une relation "DEPENDENT ON ..." qui porte sur un objet qui n'est pas un objet "role" ou un objet "aggregate" composé d'objets "role". Tous les rôles joués doivent l'être au sein du même objet "relation".

Diagnostic :

L'expression des dépendances fonctionnelles sélectionnées est incorrecte.

Correction :

Les dépendances fonctionnelles sélectionnées pour chaque objet "aggregate" doivent être supprimées ou corrigées.

tst 14-h : Sélection des objets "aggregate" composés d'objets "element" et/ou "group", pour lesquels il existe une relation "DEPENDENT ON ..." qui porte sur un objet qui n'est pas soit un objet "element","group" ou "role" soit un objet "aggregate" composé d'objets "element", "group" ou "role".
Les objets "element" qui participent à cette relation doivent tous appartenir au même objet "entity" ou "relation" et les objets "role" doivent être joués au sein du même objet "relation".

Diagnostic :

Les dépendances fonctionnelles sélectionnées sont incorrectes, une dépendance fonctionnelle doit avoir lieu au sein d'un même objet "entity" ou "relation".

Correction :

Les dépendances fonctionnelles sélectionnées pour chaque objet "aggregate" doivent être corrigées.

5.4 Description d'un scénario de contrôle a posteriori

5.4.0 Introduction

Dans ce paragraphe, nous allons commencer par décrire l'objectif du contrôle a posteriori. Ensuite comme tous les tests relatifs à la forme canonique d'un schéma e-a sont définis, nous décrirons un ordonnancement possible de ces tests.

Cet ordonnancement a pour but de regrouper les tests portant sur un même objet et ceux qui portent sur une même relation.

5.4.1 Objectif

Ce contrôle a posteriori a pour but , à partir d'un nom de schéma donné, de déterminer, dans les limites du possible, si ce schéma est sous forme canonique.

Si c'est le cas, un fichier résultat est créé dont le contenu se compose uniquement du nom du schéma.

Si le nom de schéma qui a été donné, ne correspond à aucun schéma de la base de données, le fichier résultat comportera le nom donné suivi du numéro d'erreur correspondant.

Si le schéma dont le nom est donné n'est pas sous forme canonique, alors le fichier résultat reprendra le nom du schéma traité, le nom des objets pour lesquels une ou plusieurs erreurs ont été détectées et à chaque objet repris dans le fichier, on associera les numéros d'erreur qui correspondent aux erreurs détectées.

Remarque :

Le numéro d'erreur pourrait être le numéro du test correspondant aux points 5.1, 5.2 et 5.3 .

Ces numéros d'erreur feront référence à un diagnostic et à des explications sur la ou les corrections possibles.

Dans certains cas, un nom d'objet pourra être associé au numéro d'erreur. Ce nom sera celui de l'objet qui est à l'origine de cette erreur.

ex : Si pour un objet "entity", on détecte un identifiant non minimal, on reprendra le nom de l'objet identifiant à la suite du numéro d'erreur correspondant à la violation de la règle sur les identifiants minimaux.

La structure du fichier résultat sera la suivante :

```
<nom-schéma> [ ẏ;ẏ < n°-d'erreur > ] ↵
↵
↵
↵
[ ( < nom-obj > ẏ;ẏ < type-obj> ↵
( ẏẏẏẏẏ ( < n°-erreur > [ ẏ;ẏ < nom-obj > ] [ẏ;ẏ ] )↵)
         └──────────────────────────────────────────┘
                          "A"
↵
↵ ) ]
```

Notation :

- ↵ : signifie: aller à la ligne suivante.
- [] : ce qui se trouve entre crochets est facultatif.
- () : ce qui se trouve entre parenthèses peut être répété.
- ẏ : représente un espace blanc

NB : "A" peut être répété, tant que l'on ne dépasse pas la limite de 75 caractères par ligne.

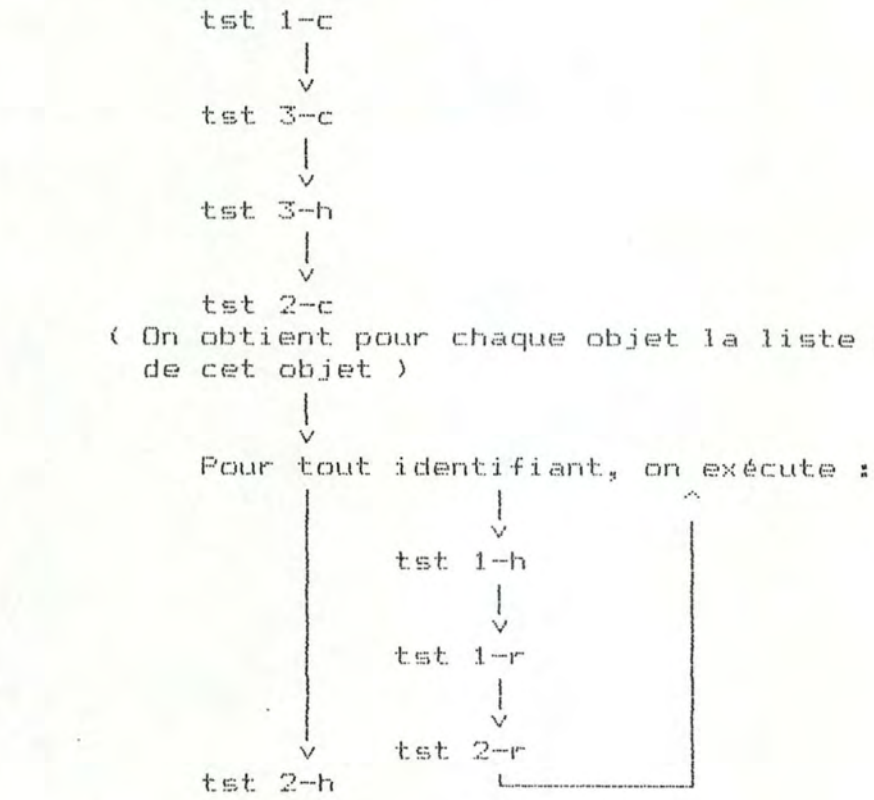
5.4.2 Ordonnancement des différents tests.

Nous allons maintenant décrire un ordre dans lequel doivent être effectués les tests.

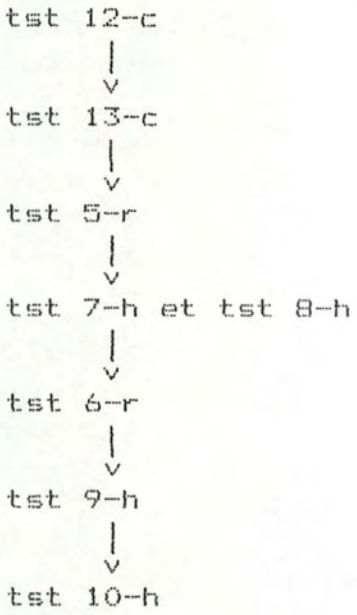
Cet ordonnancement vise à regrouper les tests par objet et par relation. Cet ordonnancement ne signifie pas que tous les objets "entity" doivent être traités en premier lieu, ensuite tous les objets "relations", etc .... L'ordre de traitement des objets pourrait être l'ordre d'apparition des objets dans les relations DSL examinées lors de l'exécution des tests par exmple.

Description de l'ordonnancement :

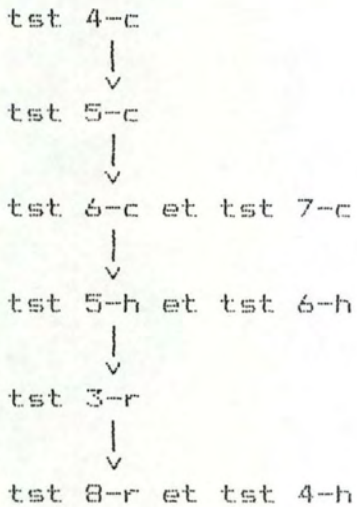
Pour les objets "entity" :



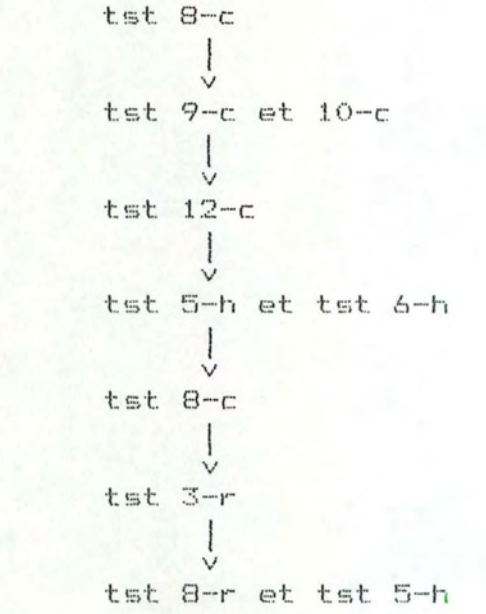
Pour les objets "relation"



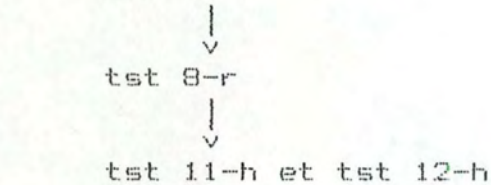
Pour les objets "element" :



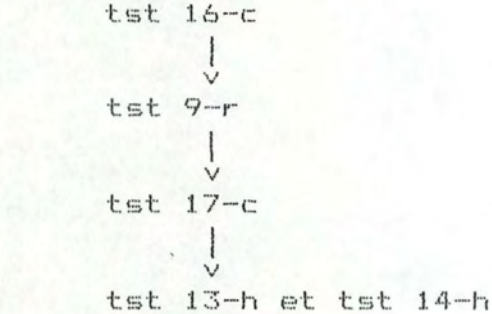
Pour les objets "group" :



Pour les objets "role" :



Pour les objets "aggregate" :



CHAPITRE 6 INTRODUCTION A L'ETUDE DU PILOTAGE INTERACTIF

6.0 Introduction

Ce dernier chapitre sera consacré à l'esquisse de la partie interactive du système de pilotage.

La réalisation d'un tel système peut être envisagée selon différentes approches :

- le pilotage passif
- le pilotage à déclenchement automatique instantané
- le pilotage à déclenchement automatique à des moments privilégiés

Nous aborderons brièvement ces différentes approches en essayant de mettre en relief leurs caractéristiques propres, leurs avantages et leurs inconvénients respectifs. Nous présenterons également les adaptations à apporter à ces approches en fonction de l'environnement dans lequel va être réalisé un tel système interactif.

Le but de cette étude n'est pas toutefois pas de choisir une approche à développer mais bien de présenter les différentes conceptions possibles avec leurs problèmes, en vue d'un développement ultérieur de ce travail.

6.1 Le pilotage passif

6.1.1 Caractéristiques

Par pilotage passif, nous entendons un système de pilotage basé sur un interface proposant diverses options à l'utilisateur, qui se voit assurer d'une liberté complète en ce qui concerne l'ordre dans lequel les spécifications doivent être introduites. D'autre part, l'utilisateur bénéficiera également d'une liberté quasi-complète dans sa volonté de déclenchement des différents tests de validation dans la mesure où les menus qui lui seront proposés, lui permettrons d'effectuer les tests qu'il désire.

Par quasi-liberté, nous voulons dire que les différents tests de validation qui sont à sa disposition (1) demandent la présence d'un certain nombre d'informations afin de s'avérer efficace. Ce qui veut dire que soit l'utilisateur devra être conscient des prérequis de ces différents tests pour les utiliser à bon escient, soit que le système lui propose les tests qui peuvent être effectués à un moment donné, mais l'utilisateur garde la liberté de les déclencher ou pas.

Quelque soit l'optique choisie, la sélection des tests nécessite une classification préalable de ces tests en fonction de l'objectif poursuivi et du type d'objet analysé comme celle qui est proposée au chapitre 5 pour la spécification des tests par exemple.

6.1.2 Avantage(s)

Ce type d'approche a l'avantage d'offrir à l'utilisateur, dans l'organisation de son travail, une liberté totale dans l'ordre de spécification des différents objets et dans la validation du schéma.

De plus, il permet à un utilisateur expert dans les propriétés que doit posséder son schéma, d'éviter le déclenchement inutile de tests, inutiles part le fait que ces tests ne détecteraient aucune violation et ne provoqueraient qu'une perte d'efficacité par accumulation des périodes d'analyse des spécifications.

---

(1) cf. Chapitre 5

6.1.3 Inconvénient(s)

Le fait de laisser à l'appréciation de l'utilisateur le déclenchement des différents tests de validation ne permet plus d'affirmer que le schéma obtenu à l'aide de ce système est sous forme canonique dans la mesure où la complétude de la validation et des corrections éventuelles qui ont été éffectuées, ne dépend pas entièrement du système.

Afin de s'assurer de la correction d'un schéma, il faudrait dès lors effectuer une validation a posteriori non interactive comme celle présentée au chapitre 5. Ce qui tendrait à faire penser que les avantages pour l'utilisateur sont en fait des obstacles à l'obtention d'un schéma valide, donc d'un système d'aide "infaillible".

6.2 Le pilotage à déclenchement automatique instantané

6.2.1 Caractéristiques

Par pilotage à déclenchement automatique instantané, nous entendons un système d'aide qui offre une souplesse totale dans l'ordre d'introduction de la spécification des objets d'un schléma e-a, telle que celle qui existe dans l'approche précédente.

Par contre, le choix du déclenchement des différents tests n'est plus laissé à l'utilisateur dans la mesure où ils sont déclenchés par le système en fonction des informations introduites par l'utilisateur et des informations nécessaires pour que les tests soient efficaces comme cela a été décrit dans le tableau de l'annexe F.

De plus, le système proposera à l'utilisateur la continuation possible du travail de spécification en cours, par l'entremise des corrections possibles à effectuer et des objets restant à spécifier, propositions qui n'altèrent en rien la liberté de l'utilisateur.

6.2.2 Avantages

Une telle approche a le double avantage d'offrir une liberté totale à l'utilisateur dans la construction de son schéma et d'assurer une validation complète du schéma par le fait que les tests sont effectués non plus de par la volonté de l'utilisateur mais sur la décision du système.

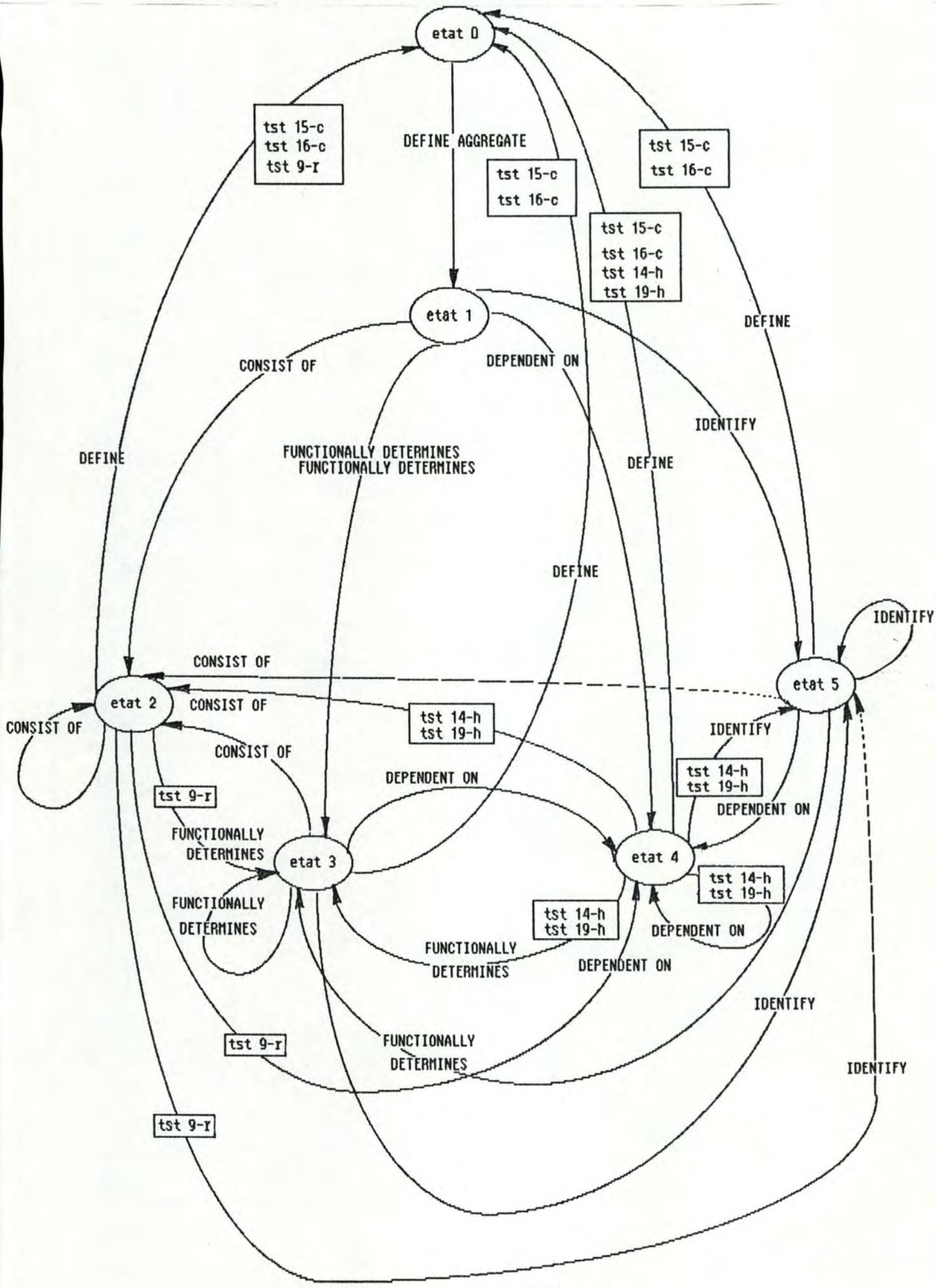
6.2.3 Inconvénients

Vu l'abscence de scénario pré-défini pour l'introduction des spécifications d'un schéma e-a, le système à réaliser devient très complexe, d'où la nécessité d'utiliser une base de connaissances qui permette d'orienter correctement l'aide à apporter. Afin de s'en persuader, il suffit simplement d'étudier quel pourrait être l'automate qui gèrerait l'aide à apporter pour la spécification d'un des types d'objet les plus simple : l'agregat.

Formalisme de représentation d'un automate :

Le scénario proposé comprend un certain nombre fini d'états (représentés par un ovale) où peuvent intervenir un ou plusieurs évènements (représentés par un texte défini sur un arc) qui sont l'introduction par l'utilisateur d'une ou plusieurs relations DSL, et où à chaque évènement correspond une action (représentée dans un rectangle) qui est l'exécution d'un ou plusieurs tests (représentés par un numéro au sein d'un rectangle) .

Diagramme de transitions d'états pour un "AGGREGATE" :



6.3 Le pilotage à déclenchement automatique à des moments privilégiés.

6.3.1 Caractéristiques

Par pilotage à déclenchement automatique à des moments privilégiés, nous entendons un système d'aide qui comporte un scénario fixe d'introduction des éléments de la spécification d'un schéma e-a. De plus, les tests sont déclenchés à certains moments privilégiés.

Le choix de ce scénario peut être fait en fonction de critères méthodologiques. Dans ce cadre, l'ordre d'introduction des éléments de spécification d'un schéma entité-association pourrait refléter une méthodologie de construction d'un schéma conceptuel afin de reproduire au sein du système de pilotage les avantages que peut apporter cette dernière.

Par exemple, il sera adopté le principe de construction progressive d'un schéma où est mis en évidence les nouveaux éléments pouvant être présents au sein de chaque proposition additionnelle prise en compte.

Ce principe conduit donc à définir un scénario tel que les types d'entité doivent être définis avant de spécifier les types d'association qui les relient. De même, ayant défini un type d'entité ou d'association, on définira ensuite les constituants qui le compose.

De plus, la spécification de contraintes d'intégrité ne pourra se faire que sur base d'éléments préalablement définis.

6.3.2 Avantage(s)

Ce type d'approche a l'avantage d'assurer une validation complète du schéma par le fait que les tests ne sont pas effectués selon la volonté de l'utilisateur mais sur la décision du système uniquement.

6.3.3 Inconvénient(s)

Vu la présence d'un scénario de spécification d'un schéma e-a, le système devient très rigide pour l'utilisateur dans la mesure où sa liberté d'organisation de sa spécification est fortement réduite.

6.4 Application à l'atelier logiciel IDA

6.4.0 Introduction

Nous allons maintenant nous resituer dans le cadre d'un poste de travail de cet atelier pour essayer de déterminer si les diverses approches abordées sont adaptables à cet environnement et de voir quelles pourraient être les éventuelles modifications à apporter soit au système d'aide, soit à l'environnement pour favoriser leur adéquation.

6.4.1 La prise en compte du poste de travail

Préliminaire :

Dans le cadre du poste de travail proposé actuellement dans l'atelier-logiciel IDA, l'utilisateur peut définir librement les différents objets d'un schéma entité-association, objets qui formeront un train d'objets contenu dans un fichier.

Etant donné cet environnement, il s'avère que les seuls moments adéquats pour le déclenchement des différents tests se situeraient après la mise-à-jour de la base de données IDA pour ce train d'objets, voire même durant la mise-à-jour pour certains tests.

Le pilotage passif :

A partir de ces quelques remarques, nous pouvons conclure que d'une part le système de pilotage passif tel qu'il est présenté au point 6.1 n'est adaptable à l'environnement actuel que par l'entremise d'une modification qui consisterait à proposer les tests uniquement aux moments adéquats définis précédemment et seulement pour les objets du train mis-à-jour.

Le pilotage à déclenchement automatique instantané :

D'autre part, l'adaptation du système de pilotage à déclenchement automatique instantané à l'environnement actuel n'est pas envisageable dans la mesure où elle nécessiterait une transformation considérable du système pour se ramener finalement à un des autres types d'approche proposés précédemment.

Par contre, une modification de l'environnement actuel pourrait être étudiée. En effet, le passage d'une base de données centralisée, comme c'est le cas actuellement, à une base de données distribuée sur les différents postes de travail pourrait constituer un nouvel environnement tout à fait adéquat à la prise en compte de ce type d'approche.

Le pilotage à déclenchement automatique à des moments privilégiés :

De même, l'adaptation du système de pilotage à déclenchement automatique à des moments privilégiés n'est possible que si l'on considère non plus ces moments privilégiés comme étant uniquement fonction des informations présentes mais en les considérant également comme étant dépendants de l'environnement, c'est-à-dire que le déclenchement des tests devra se faire durant les périodes définies dans le préliminaire.

6.4.2 Conclusion

Si l'on regarde les adaptations à apporter aux différents types de pilotage évoqués précédemment, on peut dire qu'aucun d'entre eux ne s'avère totalement adapté au poste de travail tel qu'il existe actuellement dans IDA.

Un compromis acceptable serait peut être un système de pilotage sans scénario de spécification mais où les tests de complétudes sont déclanchés lors de la mise-à-jour d'un train d'objets et les autres tests sont déclanchés selon la volonté de l'utilisateur mais toujours après la mise-à-jour d'un train d'objets.

Une autre solution, à ne pas négliger, serait d'envisager la distribution de la base de données sur les postes de travail, ce qui aurait pour avantage de permettre le développement de toutes les approches abordées au cours de ce chapitre.

Il serait peut être aussi très intéressant d'envisager la deuxième solution, puisqu'elle semble n'avoir pour inconvénient que l'absence d'une base de données distribuée.

TROISIEME PARTIE :

CONCLUSION

ET

BIBLIOGRAPHIE

CONCLUSION

L'objectif que nous nous étions fixé était de réaliser une contribution au développement d'un outil d'aide à la conception d'un schéma conceptuel d'information.

Au terme de ce travail, nous nous sommes arrêté au seuil de la spécification de l'outil de pilotage après avoir été amenés à définir les fondements d'un outil de pilotage et à prendre les options de base pour un tel outil, ce qui correspond bien à l'objectif que nous nous étions imposé.

Nous allons maintenant proposer quelques extensions à envisager, qui concernent principalement la réalisation proprement dite d'un noyau qui porte sur la mise en place de certaines classes de contrôle :

En outre,comme nous l'avons signalé au chapitre 5, un certain nombre de tests ne sont pas réalisables actuellement, comme par exemple l'étude du contenu de la relation DSL "DESCRIPTION" afin de détecter la présence d'une définition et d'une durée de vie pour certains types d'objets.

Il en est de même pour l'étude dela synonymie qui nécessite la comparaison du contenu sémantique de la définition des objets concernés.

C'est pourquoi une autre voie de recherche pourrait consister à définir une extention du langage DSL, c'est-à-dire un langage et un système d'analyse de sa sémantique, afin de permettre la réalisation de ce genre de tests.

Il serait également utile d'approfondir l'étude qui a été faite sur les possibilités d'analyse des incohérences et redondances au sein des contraintes d'intégrité, afin d'obtenir un système d'aide complet où tous les concepts du modèle e-a sont pris en compte.

De plus, comme nous l'avons déjà signalé au cours du chapitre 6, l'étude de la décentralisation de la base de données IDA vers le poste de travail constitue une extension appréciable dans la mesure où cela représenterait un apport considérable dans l'obtentiond'un environnement qui permette d'implémenter un système d'aide qui ne serait plus limité par le fait que les moments de déclenchement des tests de validation sont actuellement limités aux seuls moments de mise-à-jour des trains d'objets, mais dont les moments opportuns pourraient être définis en fonction d'une aide optimale à apporter. La décentralisation aurait donc pour conséquence de permettre la construction d'un système de pilotage beaucoup plus interactif.

Nous concluerons par la réflexion suivante :

Le système de pilotage basé sur les propriétés intrinsèques du modèle entité-association qui pourra être conçu à partir de ce travail, pourra aider l'analyste mais en aucun cas, il ne le dispensera de l'effort de créativité.

BIBLIOGRAPHIE

1) Forme générale de cette bibliographie

Les références bibliographiques, contenues dans ce rapport, se présentent sous la forme suivante :

Titre de l'article ou du livre
Auteur(s)
Maison d'édition ou Revue de parution de l'article
Année de parution

2) Classification des références bibliographiques

Les diverses références, reprises dans ce rapport, sont regroupées selon 6 classes :

. Conceptual modeling
. Entity - relationship model
. Design
. Integrity
. Relational model
. Others

2.1) Conceptual modeling

[cm.01] A NEW LOOK AT EXISTENCE DEPENDANCY IN DATA BASES
T.C. CHIANG
Proceedings AFIPS, volume 52
1983

[cm.02] ON CONCEPTUAL MODELING
J.L. BRODIE, MYLOPOULOS, J.W. SCHMIDT
Springer verslag
1984

[cm.03] ON CORRECTNESS OF INFORMATION MODELS
LUNDBERG
Information Systems, volume 8
1983

[cm.04] A FORMAL FRAMEWORK FOR DESCRIBING AND CLASSIFYING SEMANTIC DATA MODELS
STACHOWITZ
Information Systems, volume 10
1984

[cm.05] ON CONCEPTUAL DATABASE MODELING
D. MCLEOD
Acm sigmod record, volume 11, numéro 2
février 1981

[cm.06] EXTENDING THE DATABASE RELATIONAL MODEL TO CAPTURE MORE MEANING
E.F. CODD
Acm transactions on database systems, volume 2
juin 1984

[cm.07] A DECLARATIVE APPROACH TO CONCEPTUAL INFORMATION MODELING
M.R. GUSTAFSSON, T. KARLSSON, J.A. BUKENKO
Proceedings IFIP
1982

[cm.08] MODELISATION PROGRESSIVE D'UNE BASE DE DONNEES
P. DARDAILLER, C. DELOBEL, JP. GIRAUDIN
IMAG
janvier 1985

[cm.09] CADRE DE REFERENCE POUR LA CONCEPTION DE BASES DE DONNEES
JL. HAINAUT
Institut d'informatique , Namur
1983

2.2) Entity - relationship model

[er.01] TOWARD A UNIFIED VIEW OF DATA
P.P. CHEN
Acm transactions on database systems, volume 1,numéro 3
septembre 1976

[er.02] PROPERTIES OF RELATIONSHIPS AND THEIR REPRESENTATION
RAMEZ EL-MASRI, G. WIEDERHOLD
Proceeding AFIPS, volume 49
1980

[er.03] THE ENTITY-RELATIONSHIP MODEL : A BASIS FOR THE ENTERPRISE VIEW OF DATA
P. PIN, SHAN CHEN
Proceedings AFIPS, volume 46
1977

[er.04] AUTOMATIC NORMALISATION AND E-R GENERATION THROUGH ATTRIBUTES AND ROLES
K. MEYER, J. DOUGHTY
Acm sigmod record, volume 14, numéro 3
novembre 1984

[er.05] THE ENTITY-RELATIONSHIP APPROACH AS A TOOL FOR APPLICATION ANALYSIS
M.E. MODELL
The 4th international conference on entity-relationship approach
1985

[er.06] SERM : SEMANTIC ENTITY - RELATIONSHIP MODEL
M. LENZERINI
The 4th international conference on entity-relationship approach
1985

(er.07) ENHANCING THE OPERATIONAL SEMANTICS OF THE ENTITY-RELATIONSHIP MODEL
C. PARENT, S. SPACCAPIETRA
Centre de recherche en informatique de Dijon
1985

(er.08) ENTITY SETS AND THEIR DESCRIPTION
P. LINDGREEN
e-r approach to software engineering
1983

(er.09) CARDINALITY CONSTRAINS IN THE E-R MODEL .
M. LENZERINI, G. SANTUCCI
e-r approach to software engineering
1983

(er.10) CONCEPTION ASSISTEE DES APPLICATIONS INFORMATIQUES
F. BODART, Y. PIGNEUR
Presses universitaires de Namur
1983

(er.11) ERROL : EN ENTITY-RELATIONSHIP, ROLE ORIENTED, QUERY LANGUAGE
V.M. MARKOWITZ, Y. RAZ
Entity-Relationship approach to software engineering
1983

(er.12) LAMBDA : AN ENTITY-RELATIONSHIP BASED QUERY LANGUAGE FOR THE RETRIEVAL OF STRUCTURED DOCUMENTS
F. VELEZ
The 4th international conference on entity-relationship approach 1985

(er.13) DESPATH : AN ER MANIPULATION LANGUAGE
WOLFGANG ROESNER
The 4th international conference on entity-relationship approach 1985

(er.14) SCHEMA ENTITE/ASSOCIATION CANONIQUE, RESTRICTION DE LA NOTION D'IDENTIFIANT
HAINAUT JL
Institut d'informatique de Namur
Février 1986

2.3) Design

[de.01] STRUCTURED LOGICAL DESIGN OF INFORMATION SYSTEMS
BARROS, PEREZ
Information Systems, volume 5
1980

[de.02] ON CORRECTNESS OF INFORMATION MODELS
LUNDBERG
Information Systems, volume 8
1983

[de.03] FORMAL VERIFICATION OF INFORMATION DERIVABILITY IN DATABASE USING PRECEDENCE ANALYSIS
OLIVE, SALTOR
Information Systems, volume 7
1982

[de.04] A PROCEDURE TO DEFINE THE OBJECT TYPE STRUCTURE OF A CONCEPTUAL SCHEMA
VERMEIR, NIJSSEN
Information Systems, volume 7
1982

[de.05] AUTOMATIC NORMALISATION AND E-R GENERATION THROUGH ATTRIBUTES AND ROLES
K. MEYER, J. DOUGHTY
Acm sigmod record, volume 14, numéro 3
novembre 1984

[de.06] MAINTENANCE OF VIEWS
O. SCHMUELI, A. ITAI
Acm sigmod record; volume 14, numéro 2
juin 1984

[de.07] APPLICATION OF MODELING TECHNIQUES
J.W. SCHMIDT
Acm sigmod record, volume 11, numéro 2
février 1981

[de.08] ON THE DESIGN OF RELATIONAL DATABASE SCHEMATA
C. ZANIOLO, M.A. MELKANOFF
Acm transactions on database systems, volume 6,numéro 1
mars 1981

[de.09] DECOMPOSITION AND FUNCTIONAL DEPENDENCIES IN RELATIONS
W.W. ARMSTRONG, C. DELOBEL
Acm tods, volume 5, numéro 4
décembre 1980

[de.10] A FORMAL APPROACH TO THE DEFINITION AND THE DESIGN OF CONCEPTUAL SCHEMATA FOR DATABASE SYSTEMS
C. ZANIOLO, M.A. MELKANOFF
Acm tods, volume 7, numéro 1
mars 1982

[de.11] A NEW NORMAL FORM FOR THE DESIGN OF RELATIONAL DATABASE SCHEMATA
C. ZANIOLO
Acm tods, volume 7, numéro 3
septembre 1982

[de.12] STRUCTURED DATABASE SYSTEM ANALYSIS AND DESIGN THROUGH E-R APPROACH
C. HSU
The 4th international conference on entity-relationship approach 1985

[de.13] A COMPUTER-AID FOR E-R MODELING
I.T. HAWRYSZKIEWYCZ
The 4th international conference on entity-relationship approach 1985

[de.14] EASY ER : AN INTEGRATED SYSTEM FOR THE DESIGN AND DOCUMENTATION OF DATABASE APPLICATIONS
F.M. FERRERA
The 4th international conference on entity-relationship approach 1985

[de.15] DERIVATION OF ELEMENT-RELATION-ATTRIBUTE DATABASE REQUIREMENT BY DECOMPOSITION OF SYSTEM FUNCTIONS
M.W. ALFORD
e-r approach to software engineering
1983

[de.16] E-R APPROACH TO LOGICAL DATABASE DESIGN
H. SAKAI
e-r approach to software engineering
1983

[de.17] DESIGNING E-R SCHEMAS FOR CONVENTIONAL INFORMATION SYSTEMS
M.A. CASANOVA, J.E. AMARAL DE SA
e-r approach to software engineering
1983

[de.18] A COMPUTER-AIDED METHODOLOGY FOR CONCEPTUAL DATABASE DESIGN
BATINI, LENZERINI
Information Systems, volume 7
1982

[de.19] COMPUTER-AIDED ANALYSIS AND DESIGN OF INFORMATIONS SYSTEMS
NUNAMAKER J.F., KONSYNSKI B.R., HO T., SINGER C.
Communications of the ACM, volume 19, numéro 12,
1976

[de.20] EXPERT SYSTEM FOR TRANSLATING AN E-R DIAGRAM INTO DATABASES
BRIAND H., HABRIAS H., HUE J-F., SIMON Y.
The 4th international conference on entity-relationship approach 1985

[de.21] SYSTEME EXPERT EN CONCEPTION DE BASES DE DONNEES
Y. MARIE-SAINTE
LABORATOIRE INFORMATIQUE DU PROFESSEUR KOULOUMDJIAM
(mémoire DEA), Université de Lyon 1,
1985

[de.22] SECSI : SYSTEME EXPERT EN CONCEPTION DE SYSTEME D'INFORMATION
M. BOUZEGHOUD, G. GARDARIN, E. METAIS
Journée "bases de données avancées" ST-Pierre de Chartreuse, 1985

[de.23] OICSI : OUTIL INTELLIGENT DE CONCEPTION DE SYSTEME D'INFORMATION
B. FLORES, C. ROLLAND
Journée "bases de données avancées" ST-Pierre de Chartreuse, 1985

[de.24] SYCSLOG, SYSTEME LOGIQUE D'INTEGRITE SEMANTIQUE
G.T. NGUYEN, J. OLIVARÈS
Journée "bases de données avancées" ST-Pierre de Chartreuse, 1985

[de.25] A MATHEMATICAL SCHEMA FOR THE ENTITY-RELATIONSHIP DATA MODEL
H.J. SPENCER
Acm Artificial intelligence. Data Base
1985

2.4) Integrity

[in.01] THE APPLICATION OF DATATYPES TO DATA SEMANTIC INTEGRITY
BRODIE
Information Systems, volume 5
1980

[in.02] PRISM : A KNOWLEDGE BASED SYSTEM FOR SEMANTIC INTEGRITY SPECIFICATION AND ENFORCEMENT IN DATABASE SYSTEMS
A. SHELPERD, L. KERSCHBERG
Acm sigmod record, volume 14, numéro 2
juin 1984

[in.03] CONSISTENCY OF MODELS
J.M. SMITH
Acm sigmod record, volume 11, numéro 2
février 1981

[in.04] INTEGRITY CHECKING IN A LOGIC-ORIENTED E-R MODEL
R. NAKANO
3th international conference on e-r approach
1983

[in.05] SEMANTIC INTEGRITY DEPENDANCIES AND DELAYED INTEGRITY CHECKING
G.M.E. LAFUE
Proceeding of the 8th international conference on very large database, 1982

[in.06] A CONCEPTUEL MODEL FOR SEMANTIC INTEGRITY CHECKING
G.A. WILSON
Proceeding of the 6th international conference on very large database, 1980

[in.07] FAST MAINTENANCE FOR SEMANTIC INTEGRITY ASSERTIONS USING REDONDANT AGGREGATE DATA
P.A. BERNSTEIN, B.T. BLAUSTEIN, E.M. CLARKE
Proceeding of the 6th international conference on very large database, 1980

[in.08] FUNCTIONAL SPECIFICATIONS OF A SUBSYSTEM FOR DATABASE INTEGRITY
K.P. ESWAREN, D.D. CHAMBERLIN
Proceeding of the international conference on very large database 1975

[in.09] SEMANTIC INTEGRITY IN RELATIONAL DATABASE SYSTEMS
M.M. HAMMER
Proceeding of the international conference on very large database 1975

[in.10] UN OUTIL INTELLIGENT ET INTERACTIF POUR LA GESTION D'UN SYSTEME D'INTEGRITE
L. BOULANGER, J. KOULOUMDJIAM, Y. MARIE-SAINTE
Laboratoire informatique Lyon 1
1985

2.5) Relational model

[rm.01] BASES DE DONNEES ET SYSTEMES RELATIONNELS
C. DELOBEL, M. ADIBA
DUNOD

[rm.02] A STRAIGHTFORWARD FORMALIZATION OF THE RELATIONAL MODEL
NIEMI, JUERVELIN
Information Systems, volume 10
1984

[rm.03] BASES DE DONNEES : CONCEPTION ET REALISATION
A. FLORY
Economica
1982

[rm.04] DATABASE ANALYSIS AND DESIGN
ROBINSON
COMPUTER SCIENCE SERIES

[rm.05] INTRODUCTION TO DATABASE SYSTEMS
C. J. DATE
ADDISON-WESLEY

[rm.06] CONCEPTION D'UNE BASE DE DONNEES RELATIONNELLE
JL. HAINAUT
Institut d'informatique, Namur
1983

[rm.07] INTRODUCTION A UN SYSTEME DE GESTION DE BASE DE DONNEES RELATIONNELLES
JL. HAINAUT
Institut d'informatique, Namur
1983

[rm.08] A RELATIONAL MODEL OF DATA FOR LARGE SHARED DATA BANKS
CODD E.F.
Communication of ACM, volume 13, numéro 6
Juin 1970

2.6) Others

[ot.01] DATA MODELS : A SEMANTIC APPROACH FOR DATABASE SYSTEMS
A. SHELDON, BORKIN
mit press

[ot.02] PROCEEDINGS ON VERY LARGE DATA BASES
C. ZANIOLO, C. DELOBEL
Proceedings of 7th international conference on very large data bases

[ot.03] MODELING IN DATA BASE MANAGEMENT SYSTEMS
W. KENT, J.A. BUBENKO
Colloques IRIA

[ot.04] PROCEEDINGS OF INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA
B.K. KHAN, N. MINSKY, ...
Proceedings of international conference on management of data 1976

[ot.05] INTEGRITY ASPECTS OF A SHARED DATABASE
E.B. FERNANDEZ, R.C. SUMMERS
Proceedings AFIPS, volume 45
1976

[ot.06] SEMANTIC INTEGRITY ENFORCEMENT IN CENTRALIZED DBMS AND DISTRIBUTED DBMS
Z. DUSHAN, DADAL
Information Systems, volume 9
1984

[ot.07] SEMANTICS OF DATABASES : THE SEMANTICS OF DATA MODELS
BILLER, NEUMOLD
Information Systems, volume 3
1978

[ot.08] INFORMATION SYSTEMS : RECORDS, RELATION, SETS, ENTITIES AND THINGS
SENKO
Information Systems, volume 1
1975

[ot.09] A STRAIGHTFORWARD FORMALISATION OF THE RELATIONAL MODEL
T. NEIMI, K. JARVELIN
Acm sigmod record, volume 4, numéro 1
mars 1984

[ot.10] DESIGN AND IMPLEMENTATION OF AN EXTENSIBLE INTEGRITY SUBSYSTEM
E. SIMON, P. VALDURIEZ
Acm sigmod record, volume 14, numéro 2
juin 1984

[ot.11] DATA ABSTRACTIONS FOR DATABASE SYSTEMS
P.C. LOCKMANN, H.C. MAYR, W.H. WEIL, W.H. WOHLLEBER
Acm transactions on database systems, volume 4,numéro 1
mars 1979

[ot.12] COMPUTATIONAL PROBLEMS RELATED TO THE DESIGN OF NORMAL FORM RELATIONAL SCHEMAS
C. BEERI, P.A. BERNSTEIN
Acm transactions on database systems, volume 4,numéro 1
mars 1979

[ot.13] THE FUNCTIONAL DATA MODEL AND THE DATA LANGUAGE DAPLEX
D.W. SHIPMAN
Acm transactions on database systems, volume 6,numéro 1
mars 1981

[ot.14] SYNTHESIZING THIRD NORMAL FORM RELATIONS FROM FUNCTIONAL DEPENDENCIES
P.A. BERNSTEIN
Acm transactions on database systems, volume 1, numéro 4
décembre 1976

[ot.15] MULTIVALUED DEPENDENCIES AND A NEW NORMAL FORM FOR RELATIONAL DATABASES
R. FAGIN
acm transactions on database systems, volume 2,numéro 3
septembre 1977

[ot.16] NORMALIZATION AND HIERARCHICAL DEPENDENCIES IN THE RELATIONAL DATA MODEL
C. DELOBEL
Acm transactions on database systems, volume 3,numéro 3
septembre 1978

[ot.17] INDEPENDENT COMPONENTS OF RELATIONS
J. RISSANEN
Acm tods, volume 2, numéro 4
décembre 1977

[ot.18] THE THEORY OF JOINS IN RELATIONAL DATABASES
A.V. AHO, C. BEERI, J.D. ULLMAN
Acm tods, volume 4, numéro 3
septembre 1979

[ot.19] SCHEMA ANALYSIS FOR DATABASE RESTRUCTURING
SHAMKANT B. NAVATHE
Acm transactions on database systems, volume 5,numéro 2
juin 1980

[ot.20] CALCULATING CONSTRAINTS ON RELATIONAL EXPRESSIONS
A. KLUG
Acm tods, volume 5, numéro 3
septembre 1980

[ot.21] ASSOCIATIVE HARDWARE AND SOFTWARE TECHNIQUES FOR INTEGRITY CONTROL
Y.C. HONG, Y.W. STANLEY SU
Acm tods, volume 6, numéro 3
septembre 1981

[ot.22] UPDATE SEMANTICS RELATIONAL VIEWS
F. BANCILHON, N. SPYRATOS
Acm tods, volume 6, numéro 4
décembre 1981

[ot.23] FORMAL SEMANTICS FOR TIME IN DATABASES
J. CLIFFORD, S. WARREN
Acm tods, volume 8, numéro 2
juin 1983

[ot.24] THE INFORMATION RESOURCE DICTIONARY SYSTEM
GOLDFINE A.
Proceedings of the 4th international conference on e-a approach 1985

[ot.25] EXECUTABLE E-R SPECIFICATIONS FOR DATABASE SCHEMA DESIGN
S.M. STALEY, D.C. ANDERSON
Proceedings of the 4th international conference on e-a approach 1985

[ot.26] TRANSLATION OF SQL/DS DATA ACCESS/UPDATE INTO ENTITY-RELATIONSHIP DATA ACCESS/UPDATE
A.F. CARDENAS, G.R. WANG
Proceedings of the 4th international conference on e-a approach 1985

[ot.27] DISTRIBUTED DATABASE DESIGN USING AN E-R MODEL
E. BERTINO
E-r approach to software engineering
1983

[ot.28] AN IMPLEMENTATION OF A DATA DICTIONARY TO SUPPORT DATABASES DESIGNED USING THE E-R APPROACH
E-r approach to software engineering
1983

[ot.29] EXECUTABLE E-R SPECIFICATIONS FOR DATABASE SCHEMA DESIGN
SCOTT M. STALEY, DAVID C. ANDERSON
The 4th international conference on entity-relationship approach 1985

[ot.30] SEQUEL 2 : A UNIFIED APPROACH TO DATA DEFINITION, MANIPULATION, AND CONTROL
D.D. CHAMBERLIN, M.M. ASTRAHAN, K.P. ESWARAN, P.P. GRIFFITHS, ...
IBM journal research and developpment,
1976

[ot.31] Manuel de référence DSL,
Institut d'informatique (FNDP NAMUR)
1986

ANNEXES

PLAN

ANNEXE A : Transformations à apporter à un schéma entité - association non canonique.

A.1 Décomposition d'un type d'association

1) Cas où la décomposition d'un type d'association est est obligatoire :

Condition nécessaire :

Le degré du type d'association doit être supérieur à 2.

1.1) Cas simples :

1.1.1) Présence d'une seule connectivité (0-1) ou (1-1) :

Lorsqu'il existe une connectivité égale à (0-1) ou (1-1) pour un certain type d'entité, relié par un type d'association, alors il existe des dépendances fonctionnelles de ce type d' entité vers les autres types d'entité qui participent à ce type d'association.

Dès lors, le type d'association représente un certain nombre N de concepts différents qui seront représentés par N types d'association différents remplaçant le premier type d'association.

Ces N types d'association auront en commun le type d'entité dont la connectivité est (0-1) ou (1-1).

Après décomposition, la connectivité associée à chaque nom de rôle du type d'entité commun sera égale à la connectivité de celui-ci, avant décomposition, c'est-à-dire (0-1) ou (1-1).

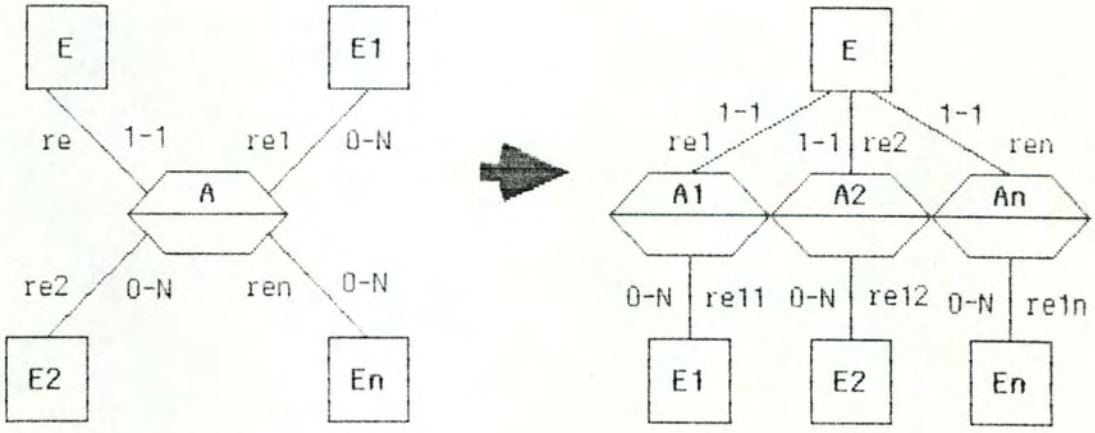
D'autre part, la connectivité associée aux noms de rôle des autres type d'entité sera égale à celle qui leur était associée avant cette décomposition.

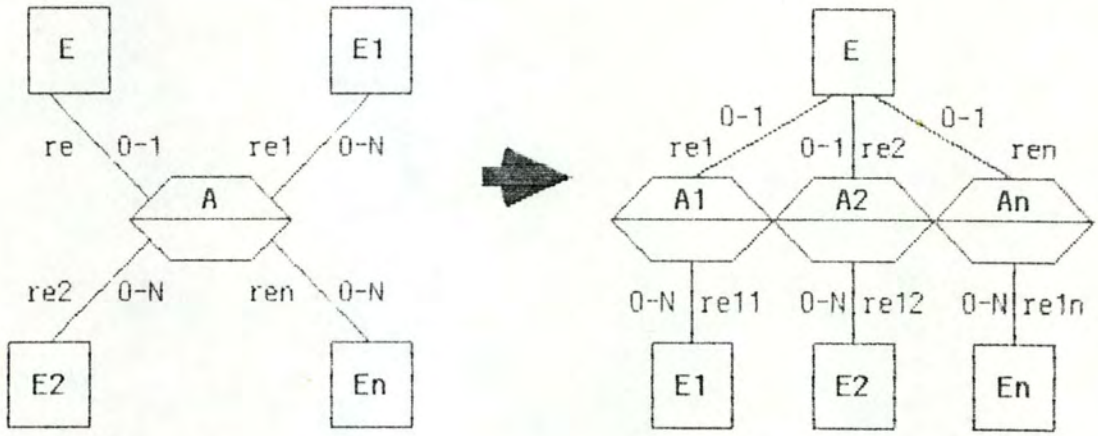
Dans le cas d'une connectivité (0-1), il sera nécessaire d'exprimer une contrainte d'égalité portant sur le type d'entité commun aux types d'association résultant de la décomposition.

Le nombre N de concepts est égal au degré du type d'association - 1.

L'identifiant des types d'association ainsi créés sera composé exclusivement du nom de rôle du type d'entité origine des dépendances fonctionnelles.

Exemple :





Contrainte d'égalité : re1 = re2 = re3

Fig. 1

1.1.2) Présence d'une dépendance fonctionnelle ou multi-valuée d'un nom de rôle vers un autre nom de rôle :

Lorsqu'il existe une contrainte d'intégrité définissant une dépendance fonctionnelle, alors on peut considérer que le type d'association représente deux concepts.
La décomposition s'effectue, dès lors de la manière suivante :

- le type d'entité déterminé de la D.F. sera retiré du premier type d'association.

- un nouveau type d'association sera défini entre les types d'entité impliqués dans la dépendance fonctionnelle.

- la connectivité du type d'entité déterminant de la D.F. , associée à son nom de rôle au sein du nouveau type d'association sera 0-1 ou 1-1 s'il était respectivement 0-? ou 1-? au sein du type d'association décomposé.

- La nouvelle connectivité du déterminé de la D.F. ou de la D.M. sera identique à l'ancienne.

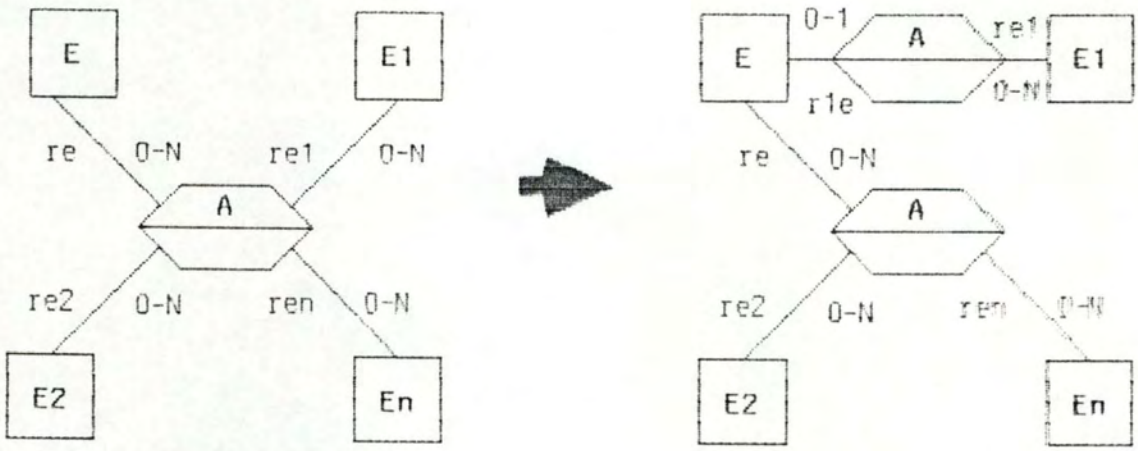
- L'identifiant du type d'association décomposé, sera modifié par suppression du nom rôle joué par le type d'entité déterminé de la dépendance fonctionnelle

- L'identifiant du type d'association ainsi créé sera composé exclusivement du nom de rôle du type d'entité déterminant de la dépendance fonctionnelle

- Une contrainte d'inclusion devra être exprimée pour les noms de rôle associés au déterminant de la dépendance pour le nouveau type d'association créé, et pour le type d'association décomposé si la connectivité associée au rôle joué par le type d'entité déterminant de la D.F. au sein du nouveau type d'association est de 0-1

- La dépendance fonctionnelle origine de cette décomposition devra être supprimée.

Exemple :



dépendance fonctionnelle : re → re1

Contrainte d'inclusion : re inclus dans r1e

Fig. 2

1.2) Cas complexes :

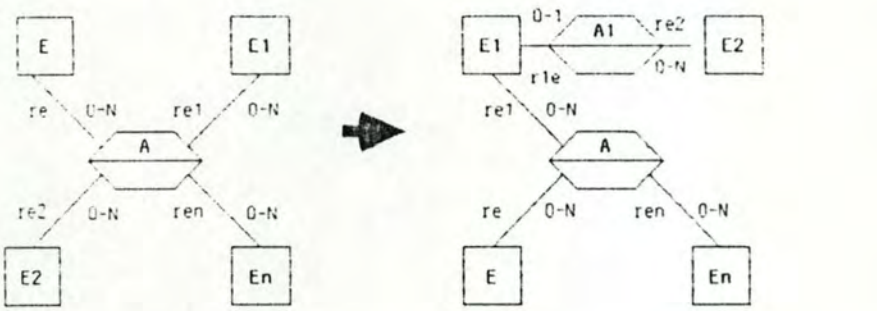
1.2.1) Présence de plusieurs dépendances fonctionnelles :

Le seul problème se posant est le choix de l'ordre de sélection des dépendances pour effectuer la décomposition.

Parmi l'ensemble de ces dépendances, il convient d'en prendre une dont le déterminé n'est pas déterminant d'une autre dépendance.
En effet, tout autre choix provoquera l'apparition de dépendance(s) fonctionnelle(s) entre rôles de types d'association différents ce que la forme canonique interdit.
Ensuite, le type d'association sera décomposé de la même manière qu'au point A.1.2, en ne tenant compte que de la dépendance ainsi choisie.

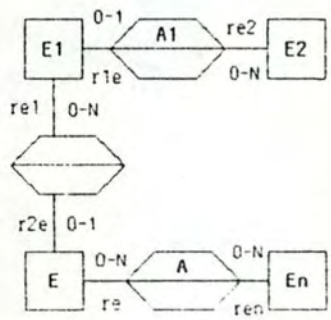
Ce processus devra être réitéré tant qu'il restera des types d'association décomposables.

Exemple :



dépendance fonctionnelle : re → re1
re1 → re2

Contrainte d'inclusion : re1 inclus dans r1e
Dépendance fonctionnelle : re → re1



Contrainte d'inclusion : re1 inclus dans r1e
re inclus dans r2e

Fig. 3

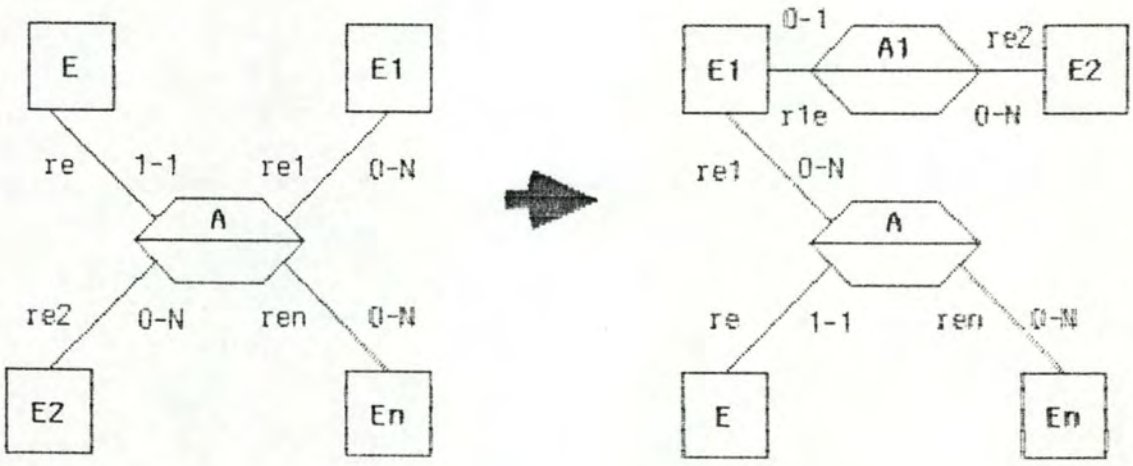
1.2.2) Présence d'une connectivité (0-1) ou (1-1) et d'une ou plusieurs dépendances fonctionnelles :

Dans ce cas, il convient de décomposer en premier lieu en fonction des dépendances.

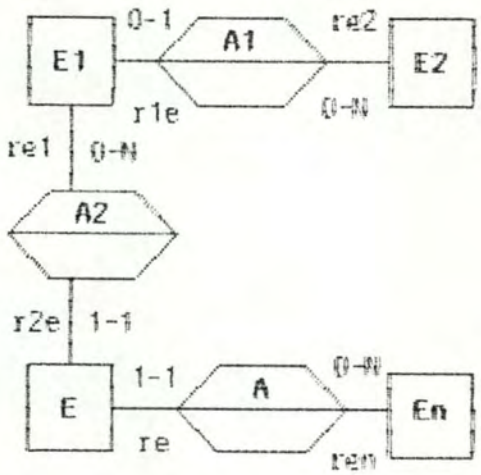
(cf. 1.1.2 ou pt 1.2.1)

Ensuite, il convient de décomposer le type d'association, résultat des décompositions successives effectuées en fonction des dépendances explicites, de la même manière que celle exprimée au point 1.1.1.

Exemple :



Dépendance fonctionnelle : re1 → re2 Contrainte d'inclusion : re1 inclus dans r1e



Contrainte d'inclusion : re1 inclus dans r1e

Fig. 4

2) Cas où la décomposition d'un type d'association est interdite :

2.1) Présence d'une dépendance fonctionnelle de plusieurs noms de rôle vers un nom de rôle

2.2) Présence de plusieurs connectivités (0-1) ou (1-1) au sein d'un même type d'association

2.3) Absence de dépendance fonctionnelle ainsi que de connectivité (0-1) et (1-1)

REMARQUE :

Dans le cas où nous sommes en présence d'un type d'association où il y a des éléments rendant obligatoire la décomposition et d'autres l'interdisant, la décomposition devra être effectuée.

Exemple :

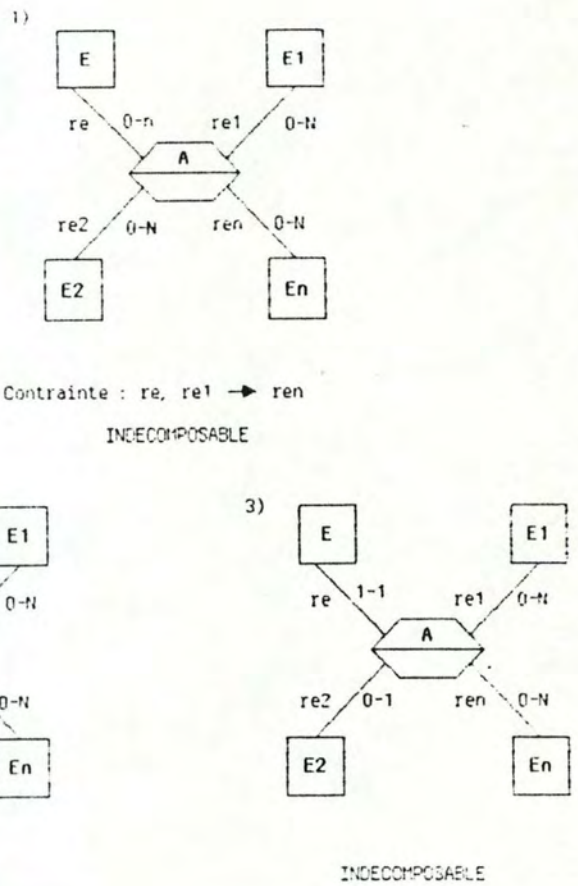


Fig. 5

A.2 Suppression d'un (groupe d') attribut(s) répétitif(s)

1. Suppression d'un attribut répétitif au sein d'un type d'entité :

   Un type d'entité possédant un attribut ou un groupe d'attributs répétitifs peut être normalisé en séparant cet attribut de ce type d'entité et en créant un nouveau type d'entité contenant cet attribut.
   Ce nouveau type d'entité ainsi créé devra respecter les propriétés de la forme canonique, ce qui veut dire par exemple qu' un identifiant devra être spécifié pour ce type d'entité.

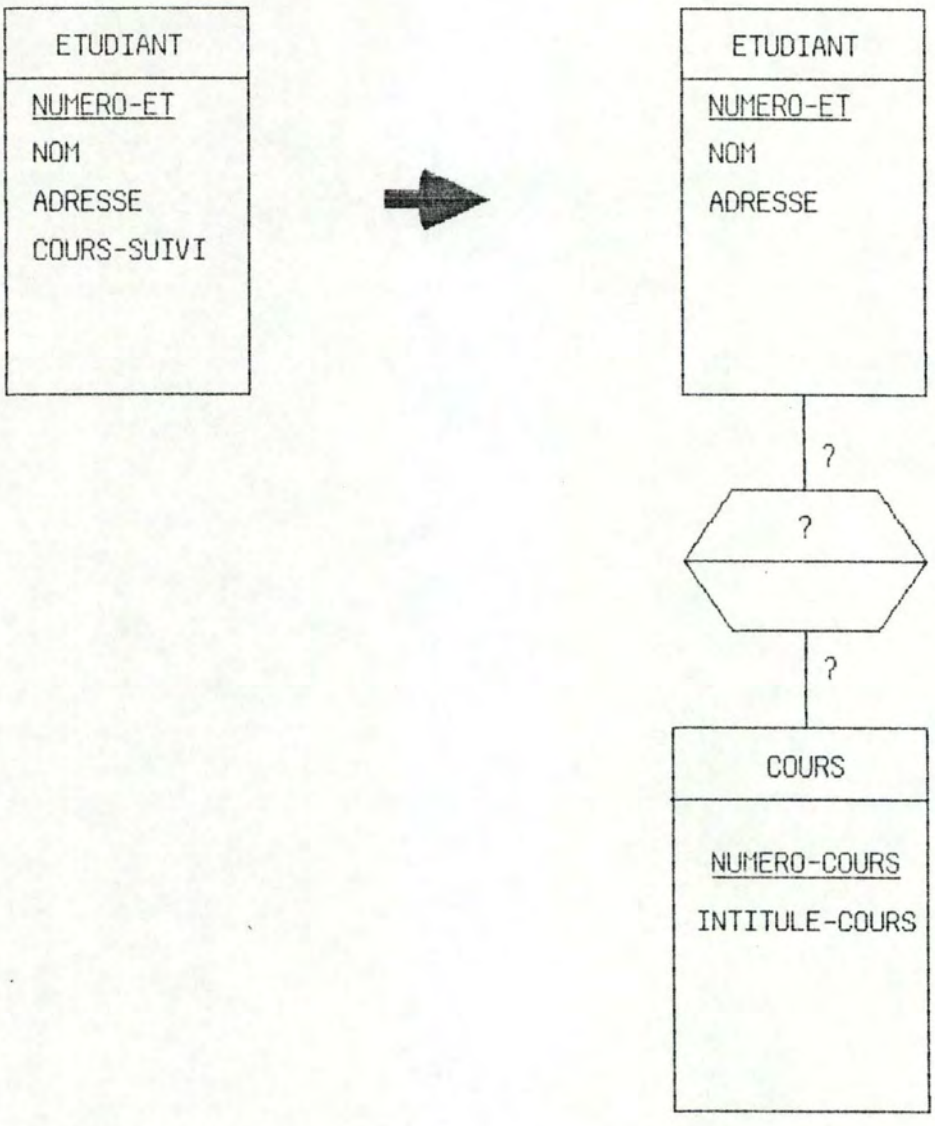
   De plus, il sera nécessaire d'expliciter le type d'association existant entre le type d'entité nouvellement créé et le type d'entité initial.

2. Suppression d'un attribut répétitif au sein d'un type d'association :

   Un type d'association possédant un attribut ou un groupe d'attributs répétitifs peut être normalisé en séparant cet attribut de ce type d'association et en créant un nouveau type d'entité contenant cet attribut.
   Ce nouveau type d'entité ainsi créé devra respecter les propriétés de la forme canonique, ce qui veut dire par exemple qu' un identifiant devra être spécifié pour ce nouveau type d'entité.

Exemple :

Soit un type d'entité ETUDIANT possédant un attribut décomposable répétitif : cours—suivi, composé des attributs : . numéro de cours
. intitulé—cours



? signifie : à déterminer par l'utilisateur

FIG. 6

A.3 Modifications à apporter lorsqu'un (groupe d')attribut(s) ne dépend que d'une partie de l'identifiant

1. Si un attribut d'un type d'entité ne dépend que d'une partie de la clé :

   Il sera nécessaire de créer un nouveau type d'entité ayant pour identifiant la partie de la clé qui identifie cet attribut et comme autre attribut, le dernier cité, ainsi qu'un type d'association reliant ce nouveau type d'entité à l'ancien.

   Ce nouveau type d'entité devra respecter les propriétés de la forme canonique.

   Le type d'entité initial se verra enlevé cet attribut, ainsi que la partie de la clé en question.
   La partie enlevée de la clé sera remplacée par le nom de rôle du nouveau type d'entité au sein du nouveau type d'association créé.

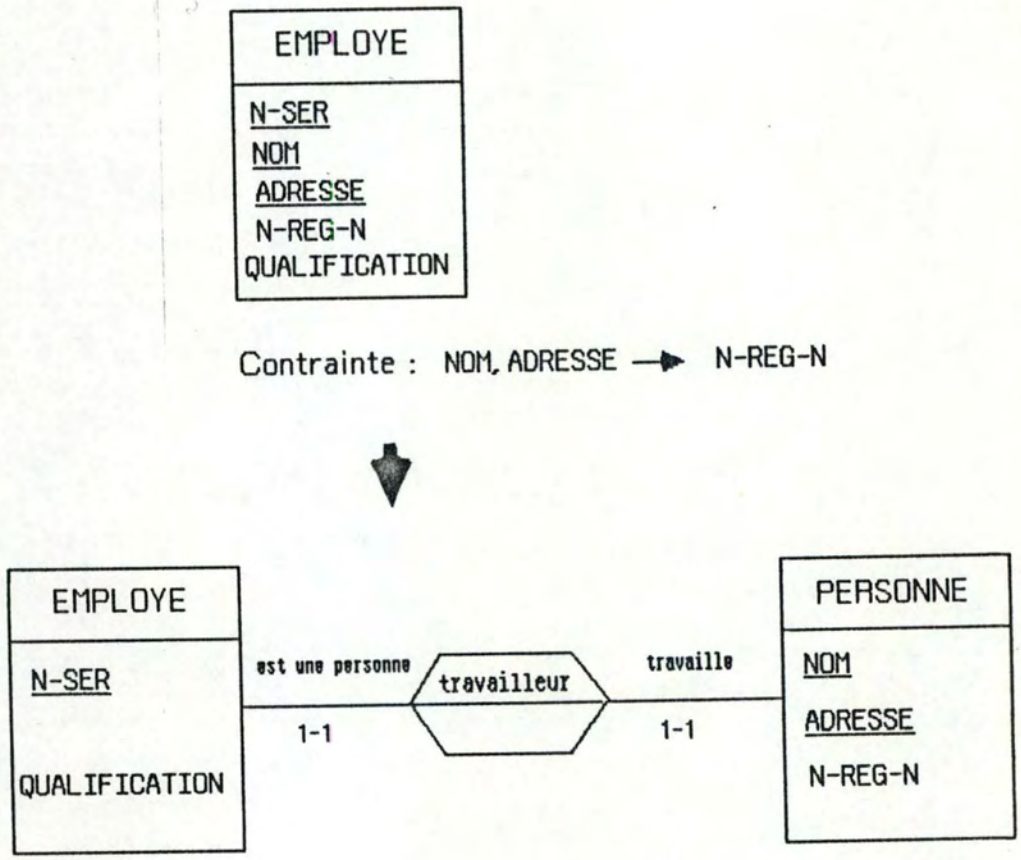


Fig. 9

Contrainte : L'identifiant d'"EMPLOYE" est composé de "N-SER" et de "travaille"

A.4 Construction d'un identifiant minimal

1. Identifiant minimal d'un type d'entité :

Si un élément d'un identifiant est déterminé fonctionnellement par un autre élément de cet identifiant, il convient de supprimer cet élément de cet identifiant.

REMARQUE :

Une fois, cet élément sorti de l'identifiant, il convient de s'assurer que l'on ne se trouve pas dans les conditions évoquées à l'annexe A.3.

Exemple :

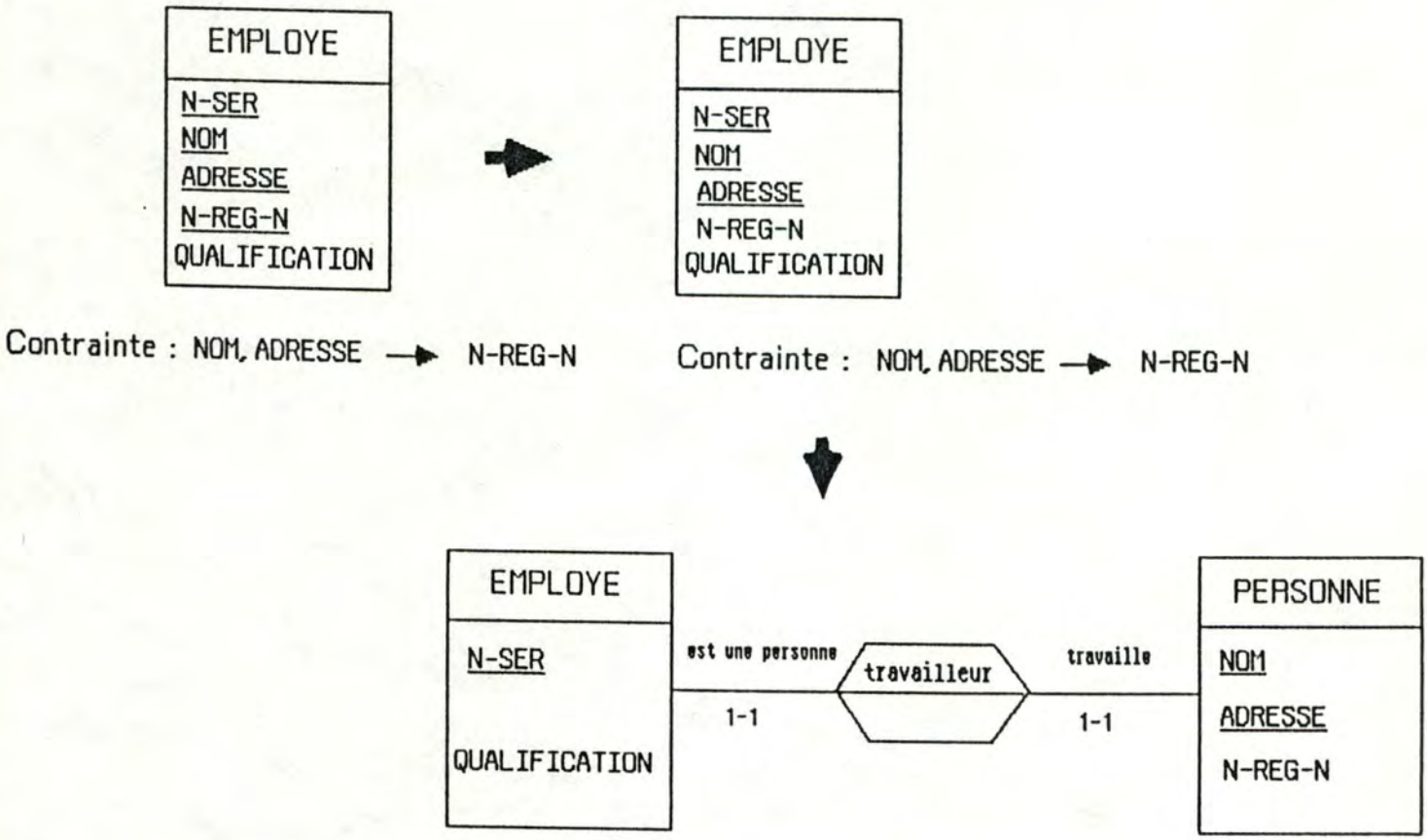
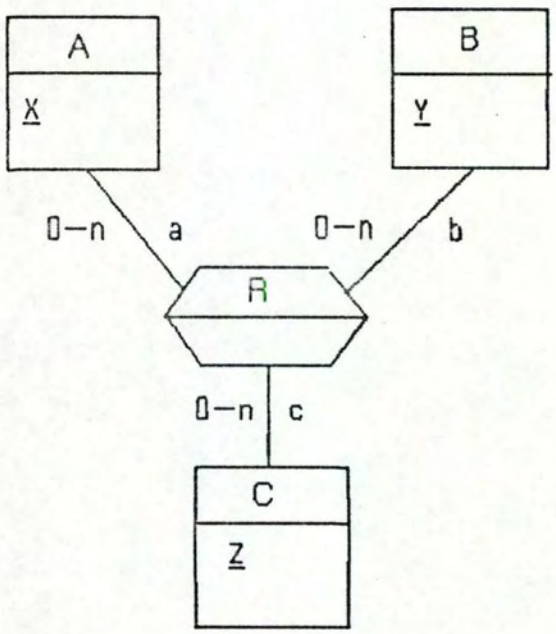


Fig. 8

2. Identifiant minimal d'un type d'association :

Dans le cas où il existe une dépendance fonctionnelle d'un ou de plusieurs éléments éléments vers un autre et si l'association n'est plus décomposable, il faut supprimer de l'identifiant l'élément déterminé.

exemple :



Contrainte : a,b → c

Remarque : L'identifiant de R est : a,b et non a,b,c

ANNEXE B : Représentation d'un schéma entité - association en DSL.

Cette annexe reprend les différentes relations dsl qui peuvent être utilisées pour spécifier les différents concepts d'un schéma entité - association.

1.1 Objet "entity"

La spécification d'un objet "entity" peut, en général, comporter les éléments suivants :

a) Le nom de l'objet "entity" :

```
DEFINE ENTITY entity-name ;
```

b) Synonyme(s) :

```
SYNONYM (synonym-name [,]) ;
```

c) Description :

```
DESCRIPTION ;
              DEF : TEXT .
              DURATION : TEXT;
```

d) Ses éventuels constituants (0 à N) :

```
CONSIST OF ( { syst-par-name } { group-name   } ) ;
           { integer      } { element-name }
```

e) Ses éventuels identifiants (1 à M) :

```
IDENTIFIED BY { group-name     } ;
              { element-name   }
              { role-name      }
              { aggregate-name }
```

f) Cardinalité : nombre maximum d'occurrences de l'objet "entity" à une période donnée

```
CARDINALITY IS { sys-par-name } IN set-name
            [ DURING { calendar-name } ] ;
                     { string        }
```

1.2 Objet "element" / "group"

La spécification d'un objet "element" ou "group" peut, en général, comporter les éléments suivants :

a) Le nom de l'objet :

```
DEFINE ELEMENT element-name ;
DEFINE GROUP group-name ;
```

b) Synonyme(s) :

```
SYNONYM (synonym-name [,] ) ;
```

c) Description :

```
DESCRIPTION ;
            DEF : TEXT.
            DURATION : TEXT ;
```

d) Soit l'objet "entity", soit l'objet "relation", soit le ou les objets "group" que l'objet "element" décrit.

```
CONTAINED ( [ { sys-par-name } TIMES ]
              { integer      }

          IN { entity-name   } [,] ) ;
             { relation-name }
             { group-name    }
```

e) Format de l'objet "element" :

```
FORMAT IS string ;
```

f) Domaine de valeurs de l'objet "element" :

soit

```
DOMAIN OF VALUE ARE ( { sys-par-name }
                      { any-value    }

     [ THRU { sys-par-name } ] ) ;
            { any-value    }
```

soit

```
SAME DOMAIN AS element-name ;
```

g) Simple ou répétitif :

Cette caractéristique est exprimée dans la première partie optionnelle de la clause suivante :

```
CONTAINED ( [ { sys-par-name } TIMES ]
              { integer      }

          IN { entity-name   } [,] ) ;
             { relation-name }
             { group-name    }
```

h) Obligatoire ou facultatif :

Cette caractéristique est exprimée dans la première partie optionnelle de la clause suivante :

```
CONTAINED ( [ { sys-par-name } TIMES ]
              { integer      }

          IN { entity-name   } [,] ) ;
             { relation-name }
             { group-name    }
```

En effet, si l'objet "element" / "group" est contenu au moins une fois au sein d'un autre objet, il sera considéré comme obligatoire.
Dans le cas contraire, il sera considéré comme facultatif.

i) Probabilité d'avoir une valeur nulle :

Cette caractéristique peut être exprimée au moyen de la clause suivante, et ce, grâce au systeme-parameter :

```
DOMAIN OF VALUE ARE ( { sys-par-name }
                      { any-value    }

      [ THRU { sys-par-name } ] ) ;
             { any-value    }
```

j) Les constituants de l'objet "group" :

```
CONSIST OF ( { syst-par-name } { group-name   } ) ;
             { integer       } { element-name }
```

1.4 Objet "relation"

La spécification d'un objet "relation" peut, en général, comporter les éléments suivants :

a) Le nom de l'objet "relation" :

```
DEFINE RELATION relation-name ;
```

b) Synonyme(s) :

```
SYNONYM (synonym-name [,]) ;
```

c) Description :

```
DESCRIPTION ;
            DEF : TEXT.
            DURATION : TEXT ;
```

d) Pour chaque role joué au sein du type d'association :

```
RELATES entity-name [ AS role-name ]
        [ WITH CONNECTIVITY { sys-par-name } ] ;
                            { string       }
```

e) Cardinalité :

```
CARDINALITY IS { sys-par-name } IN set-name
               { integer      }

               [ DURING { calendar-name } ] ;
                        { string        }
```

f) Attributs :

```
CONSIST OF ( [ { sys-par-name } ] { group-name   }[,]);
             { integer      }     { element-name }
```

g) Identifiant :

```
IDENTIFIED BY { role-name      } ;
              { aggregate-name }
```

1.5 Objet "role"

La spécification d'un objet "role" peut, en général, comporter les éléments suivants :

a) Le nom de l'objet "role" :

```
DEFINE ROLE role-name ;
```

b) Les synonymes possibles pour cet objet "role" :

```
SYNONYM ( synonym-name [,] ) ;
```

c) Une description de cet objet "role".

```
DESCRIPTION ;
            TEXT ;
```

d) Le nom de l'objet "entity" qui joue ce rôle,
le nom du type d'association au sein duquel est joué ce rôle,
la connectivité associée à ce rôle :

```
ASSUMED BY entity-name IN relation-name

        [ WITH CONNECTIVITY { sys-par-name } ] ;
                            { string       }
```

1.6 Objet "aggregate"

La spécification d'un objet "aggregate" peut, en général, comporter les éléments suivants :

a) Le nom de l'objet "aggregate" :

```
DEFINE AGGREGATE aggregate-name ;
```

b) Les synonymes possibles pour cet objet :

```
SYNONYM ( synonym-name [,] ) ;
```

c) Une description de cet objet :

```
DESCRIPTION ;
            TEXT ;
```

d) Ses constituants :

```
INCLUDES { group-name     } ;
         { element-name   }
         { role-name      }
         { aggregate-name }
```

e) Sa fonction c'est-à-dire soit l'objet "entity" ou "relation" qu'il identifie, soit le déterminant ou le déterminé de la dépendance fonctionnelle dans laquelle il joue un rôle :

soit

```
IDENTIFIES { entity-name   } ;
           { relation-name }
```

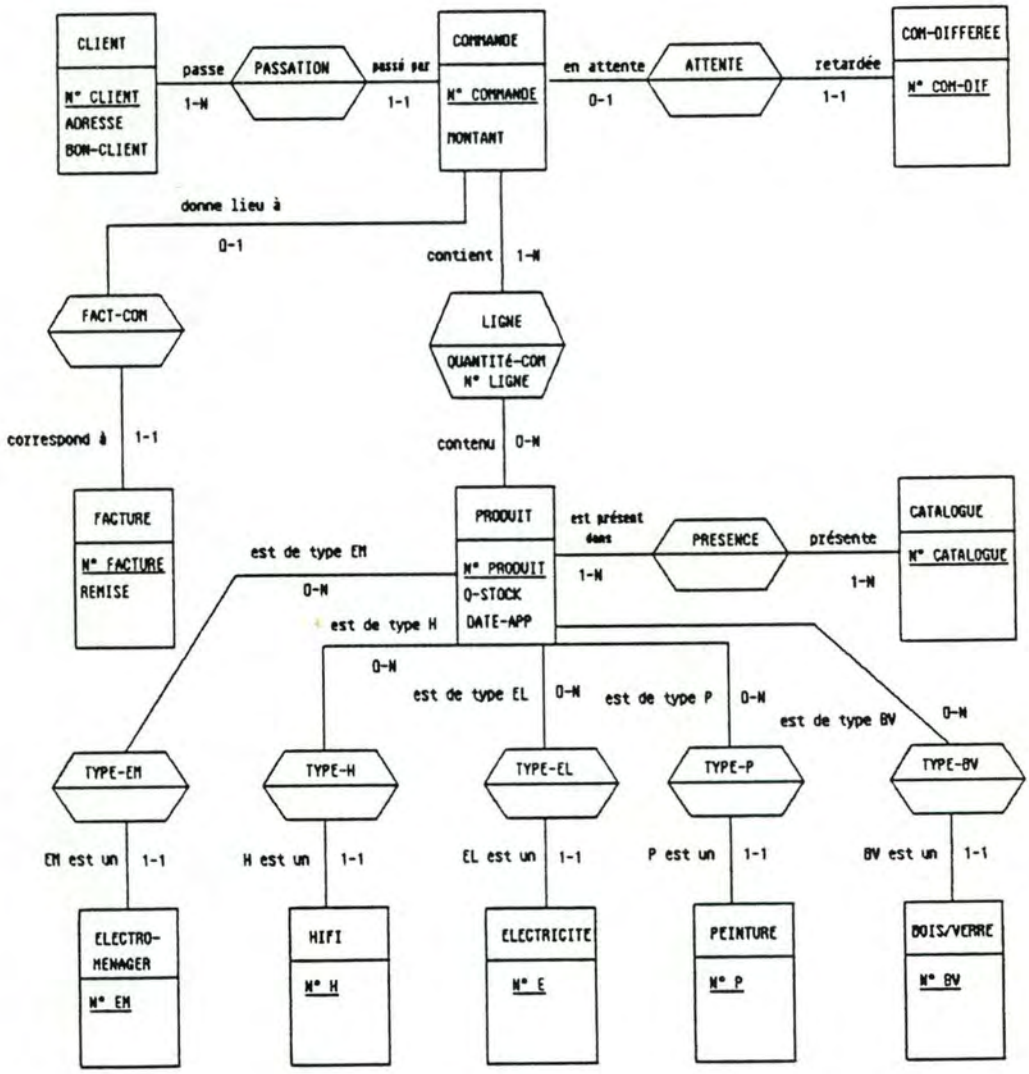
soit

```
FUNCTIONALLY DETERMINES { element-name   }
                        { group-name     }
                        { role-name      }
                        { aggregate-name }

             FOR { entity-name   } ;
                 { relation-name }
```

soit

```
FUNCTIONALLY DEPENDENT ON  { element-name   }
                           { group-name     }
                           { role-name      }
                           { aggregate-name }

             FOR { entity-name   } ;
                 { relation-name }
```

ANNEXE C : Contraintes d'intégrité : exemple.



ANNEXE D : Extracteur de base de données IDA :

Spécification & contraintes

D.1 Spécification de l'extracteur

En guise de spécification de cet extracteur, nous allons spécifier les propriétés des entrées et sorties de cet extracteur :

Les entrées :

Dans un premier temps, nous supposerons qu'un seul "set" de nom donné peut être extrait. Par la suite, une extension pourra être envisagée. Cette extension permettrait à l'utilisateur de déterminer les éléments d'un schéma qu'il souhaite extraire de la base de données.

Nous formulons également une contrainte à propos du "set" extrait :

Ce "set" ne pourra pas contenir la notion de "subset. On ne pourra extraire qu'un set composé uniquement d'objets "entity", "relation", "element", "group", "role".

Nous émettons également l'hypothèse que tout schéma extrait répond à la définition de la forme canonique d'un schéma entité-association DSL. Si cette condition n'est pas vérifiée, nous ne pouvons garantir une extraction correcte de ce schéma.

Les sorties :

I Présentation de la structure des fichiers.

1) Introduction

Nous avons opté pour une séparation des informations en deux parties, car certaines d'entre elles seront utilisées pour la création de bases de données, les autres n'étant reprises qu'à des fins documentaires. C'est pourquoi, nous obtiendrons, en résultat de l'extraction, un fichier principal, contenant les informations "utiles" et un fichier auxiliaire reprenant toutes les informations considérées comme documentation, si aucune information ne peut être extraite, les fichiers ne comprendront qu'une seule ligne, celle donnant la date d'extraction et le nom du demandeur.

Dans ce paragraphe, nous définirons les différents éléments repris au sein des fichiers principal et auxiliaire, ainsi que leur syntaxe . Ensuite nous établirons la structure de ces fichiers.

2) Définition de la syntaxe utilisée

Nous avons opté pour une séparation des informations en deux parties, car certaines d'entre elles seront utilisées pour la création de bases de données, les autres n'étant reprises qu'à des fins documentaires. C'est pourquoi, nous obtiendrons, en résultat de l'extraction, un fichier principal, contenant les informations "utiles" et un fichier auxiliaire reprenant toutes les informations considérées comme documentation, si aucune information ne peut être extraite, les fichiers ne comprendront qu'une seule ligne, celle donnant la date d'extraction et le nom du demandeur.

() expriment la répétitivité de l'objet qui est compris entre ces parenthèses.

; séparateur unique entre 2 éléments.

↵ signifie qu'un "line feed" est obligatoire.

Les mots en majuscule sont des noms repris tels quels dans les fichiers.
Tout ce qui apparait en majuscule dans la description des fichiers, pour plus de clarté, sera écrit en minuscule dans les fichiers en sortie de l'extracteur.

Tout ce qui est en caractères gras sera, obligatoirement, présent dans le fichier correspondant si l'objet dans lequel est repris la valeur, est présent.
L'obligation pour un objet d'être présent dans le fichier, correspond à l'existence obligatoire ou non de cet objet dans la forme canonique DSL.

L'absence d'une valeur, non obligatoire, se marquera par la présence de deux séparateurs consécutifs.

Si sur une ligne, aucune valeur, correspondant aux différents éléments de cette ligne, n'est présente, alors cette ligne ne sera pas présente dans le fichier.

REMARQUE:

A l'intérieur d'un objet "entity", "group", "element","relation", "role", "identifier", "dep-fct", l'ordre des phrases (ou ligne) n'a pas d'importance.

Seuls les blancs significatifs, c'est-à-dire ceux présents dans les valeurs, seront recopiés dans les fichiers. Dans la structure des fichiers, décrite par la suite, les blancs qui encadrent les labels, ne sont présents que pour permettre une présentation plus claire.

3) Définition des éléments de base

<texte> ::== Suite illimitée de caractères

<chaîne x> ::== Suite de x caractères au plus

<nombre> ::== <chaîne 10>

<moyenne> ::== <nombre>

<min / max> ::== <nombre>

<niveau> ::== < chaîne 2>

c'est-à-dire : un chiffre de 00 à 10

<date-d-maj / date-extraction> ::== <chaine 6>

c'est-à-dire : AAMMJJ (inverse de l'ordre DSL)

<prob> ::== <nombre>

<nom-ent / nom-rel / nom-el / nom-gr / nom-rol> ::== <chaîne 30>

<nom-demandeur / nom-s> ::== <chaîne 30>

<type-form / type-dis> ::== <chaîne 30>

<nbre-av-virgule > ::== <chaîne 5>

<nbre-ap-virgule> ::== <chaîne 3>

<sequence> ::== <chaîne 1>

c'est-à-dire : Si nous sommes en présence d'une valeur unique, la séquence vaudra 0.

Si nous sommes en présence d'un intervalle, séquence vaudra 1 pour la ligne contenant le minimun et 2 pour la ligne comprenant le maximun. Ces deux lignes devront être consécutives dans le fichier.

<code> ::== <chaîne 30>

c'est-à-dire : la valeur du code dans la clause " CODE ..."

<valeur> ::== <chaîne 30>

c'est-à-dire : soit une des valeurs du domaine de valeurs

soit une valeur attribuée à un code donné

<paramètre> ::== <nombre>;<nombre>

<num-ord> ::== <nombre>

c'est-à-dire : un entier représentant un numéro d'ordre pour les objets "element" et "group" à l'intérieur de l'objet au quel ils appartiennent.

<nom-at> ::== <chaîne 30>

c'est-à-dire : le nom d'un objet "attribute"

REMARQUE :

Tout nombre verra sa valeur cadrée à gauche.

4) Présentation de la structure du fichier principal et du fichier auxiliaire.

4.1) Introduction

Dans cette quatrième partie, nous décrivons la structure des fichiers de sortie de l'extracteur de la base de données IDA.

La structure de ces deux fichiers sera la suivante :

Nous trouverons en entête de chaque fichier, la date de l'extraction ainsi que le nom du demandeur.

Dans le fichier principal, nous trouveronsla spécification d'un set, suivie par l'ensemble des entités lui appartenant. Chaque entité est suivie par l'ensemble de ses constituants, c'est-à-dire des élément(s), des groupe(s) et leurs constituants. Ensuite, nous trouvons les associations, chacune étant accompagnée de ses constituants et des rôles joués au sein de celle-ci. Ce n'est qu'ensuite qu'apparaîtront l'ensemble des identifiants et l'ensemble des dépendances fonctionnelles.

Afin de faciliter la compréhension de la structure des constituants d'une entité ou d'une association, un numéro de niveau sera attribué à chaque constituant, numéro qui reflètera le niveau d'appartenance à la structure de l'entité ou de l'association (arborescence) :

- Tout élément ou groupe qui appartient directement à l'entité ou l'association, c'est-à-dire apparaissant dans la relation "consist of" sera précédé du niveau 01
- Tout élément ou groupe apparaissant dans la relation "consist of" d'un groupe, sera précédé d'un niveau supérieur d'une unité par rapport au niveau du groupe le contenant.

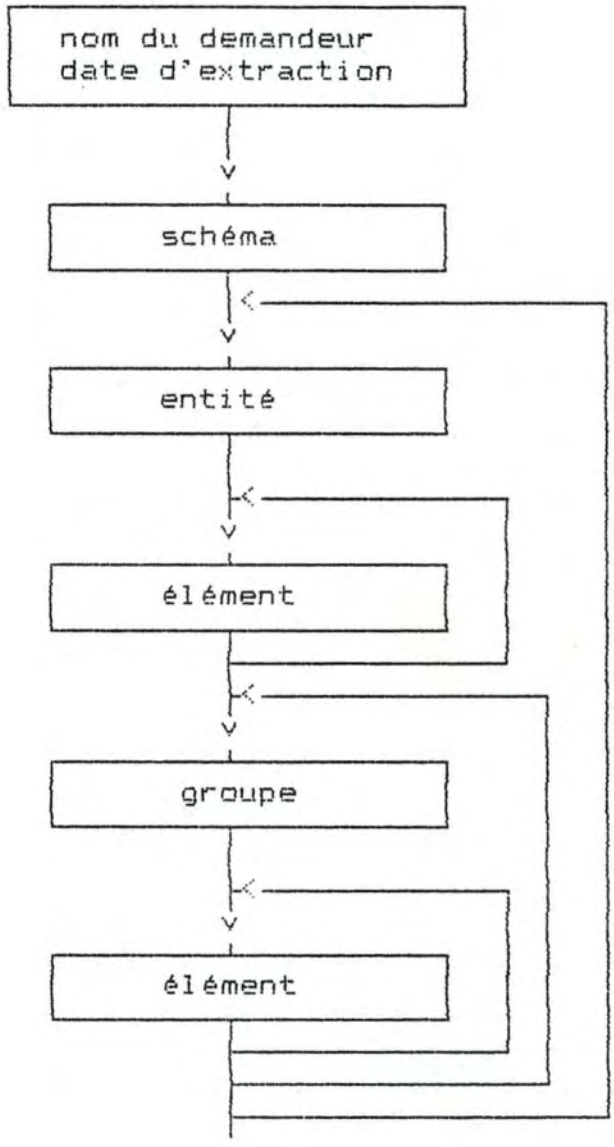
Un numéro supplémentaire sera également repris, celui-ci indiquera quel est le numéro d'ordre du constituant au sein de l'objet auquel il appartient. Dans le second fichier, nous trouverons des informations supplémentaires pouvant intéresser l'utilisateur c'est-à-dire la description et les synonymes de chaque objet se trouvant dans le premier fichier, ainsi que le domaine de valeurs et la table de codification des éléments. Ce fichier suit la même structure que le précédent.

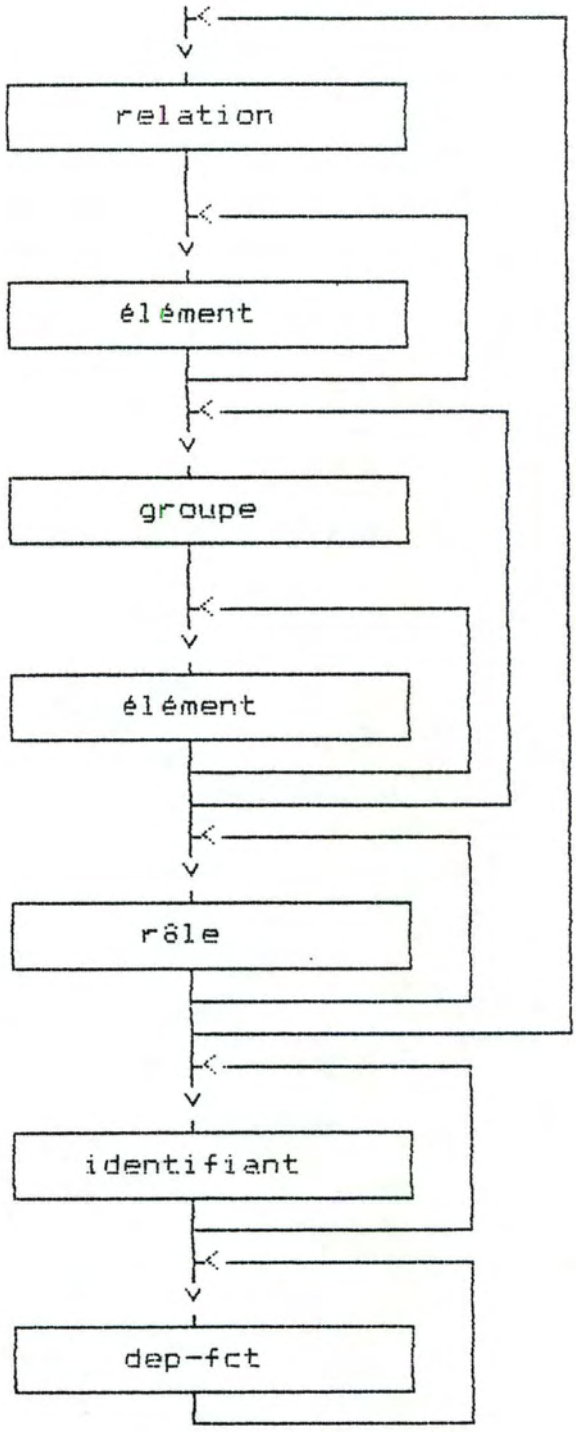
4.2) Description de la structure

a) pour le fichier principal

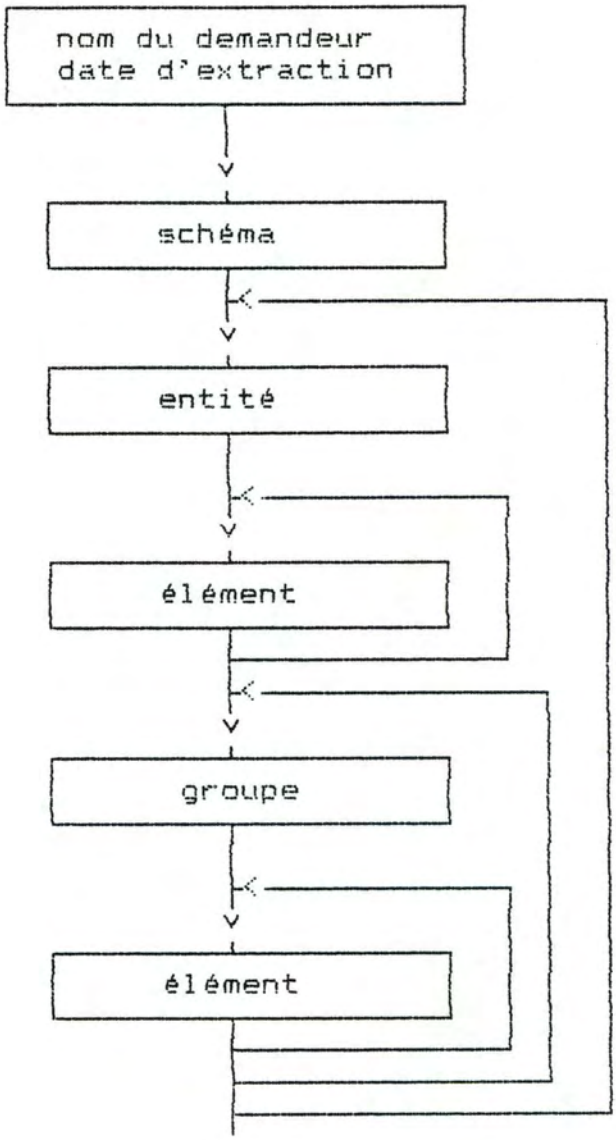
REMARQUE :

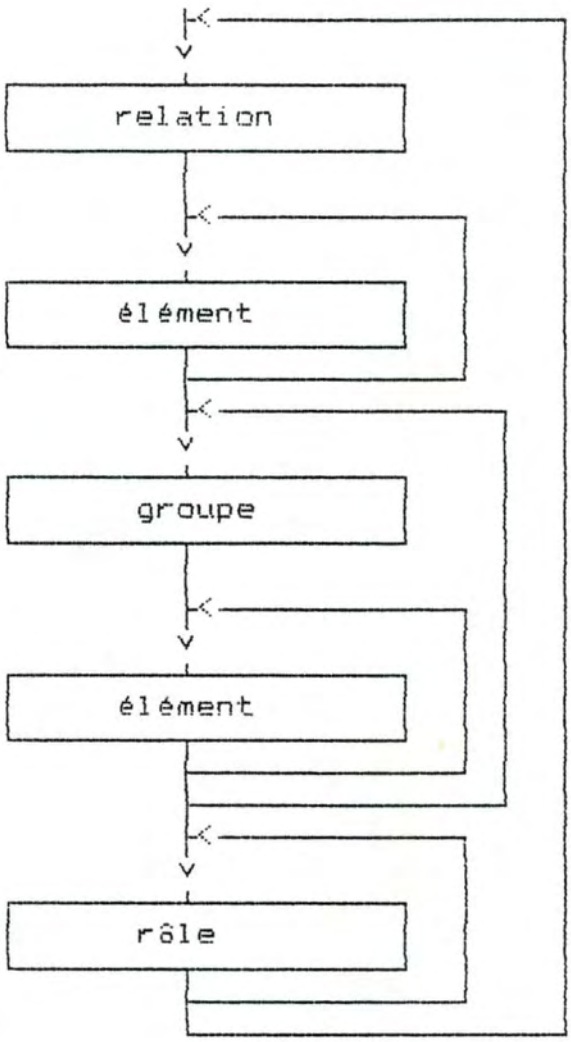
——> signifie : "est suivi de"




relation
élément
groupe
élément
rôle
identifiant
dep-fct

b) pour le fichier auxiliaire




relation
élément
groupe
élément
rôle

4.3) Description du contenu des deux fichiers

a) Le fichier principal

SYSTEM; <nom-demandeur>;<date-extraction>;↵

SCHEMA; <nom-s>;<date-d-maj>;↵

(ENTITY; <nom-ent>; <date-d-maj>;↵

(IMAGE;<type>;<nom>;↵)

(ATTRIBUTE;<nom-at>;↵

(STATISTICS;AT;VAL;<chaîne 30>;↵)

(STATISTICS;AT;RAN;<min>;<prob>;<max>;↵)

STATISTICS;AT;DIS;<typ-dist>;<paramètre>;↵

)

STATISTICS; CARD; VAL; <min>;<prob>;<max>;<moyenne>;↵

(STATISTICS; CARD; RAN; <min>;<prob>;<max>;;↵)

STATISTICS; CARD; DIS; <typ-dist>;<paramètre>;↵

SUB_TYPE; <nom-ent>;<weak/strong>;↵

(ELEMENT; <niveau>;<num-ord>;<nom-el>;<date-d-maj>;<typ-form>;
<nbre-av-virgule>;<nbre-ap-virgule>;↵

(IMAGE;<type>;<nom>;↵)

(ATTRIBUTE;<nom-at>;↵

(STATISTICS;AT;VAL;<chaîne 30>;↵)

(STATISTICS;AT;RAN;<min>;<prob>;<max>;↵)

STATISTICS;AT;DIS;<typ-dist>;<paramètre>;↵

)

STATISTICS; REPET; VAL;<min>;<prob>;<max>;<moyenne>;↵

(STATISTICS; REPET; RAN; <min>;<prob>;<max>;;↵)

STATISTICS; REPET; DIS; <typ-dist>;<paramètre>;↵

)

```
( GROUP; <niveau>;<num-ord>;<nom-gr>;<date-d-maj>;↵
    (IMAGE;<type>;<nom>;↵)
    (ATTRIBUTE;<nom-at>;↵
        (STATISTICS;AT;VAL;<chaîne 30>;↵)
        (STATISTICS;AT;RAN;<min>;<prob>;<max>;↵)
        STATISTICS;AT;DIS;<typ-dist>;<paramètre>;↵
    )
    STATISTICS; REPET; VAL; <min>;<prob>;<max>;<moyenne>;↵
    (STATISTICS; REPET; RAN; <min>;<prob>;<max>;;↵)
    STATISTICS; REPET; DIS; <typ-dist>;<paramètre>;↵
    ( ELEMENT;<niveau>;<num-ord>;<nom-el>;<date-d-maj>;<typ-form>;
            <nbre-av-virgule>;<nbre-ap-virgule>;↵
        (IMAGE;<type>;<nom>;↵)
        (ATTRIBUTE;<nom-at>;↵
            (STATISTICS;AT;VAL;<chaîne 30>;↵)
            (STATISTICS;AT;RAN;<min>;<prob>;<max>;↵)
            STATISTICS;AT;DIS;<typ-dist>;<paramètre>;↵
        )
        STATISTICS; REPET; VAL;<min>;<prob>;<max>;<moyenne>;↵
        (STATISTICS; REPET; RAN; <min>;<prob>;<max>;;↵)
        STATISTICS; REPET; DIS; <typ-dist>;<paramètre>;↵
    )
)
```

```
( RELATION; <nom-rel>; <date-d-maj>;↵
    (IMAGE;<type>;<nom>;↵)
    (ATTRIBUTE;<nom-at>;↵
         (STATISTICS;AT;VAL;<chaîne 30>;↵)
         (STATISTICS;AT;RAN;<min>;<prob>;<max>;↵)
         STATISTICS;AT;DIS;<typ-dist>;<paramètre>;↵
    )
    STATISTICS; CARD; VAL; <min>;<prob>;<max>;<moyenne>;↵
    (STATISTICS; CARD; RAN; <min>;<prob>;<max>;;↵)
    STATISTICS; CARD; DIS; <typ-dist>;<paramètre>;↵
    ( ELEMENT; <niveau>;<num-ord>;<nom-el>;<date-d-maj>;<typ-form>;
              <nbre-av-virgule>; <nbre-ap-virgule>;↵
         (IMAGE;<type>;<nom>;↵)
         (ATTRIBUTE;<nom-at>;↵
              (STATISTICS;AT;VAL;<chaîne 30>;↵)
              (STATISTICS;AT;RAN;<min>;<prob>;<max>;↵)
              STATISTICS;AT;DIS;<typ-dist>;<paramètre>;↵
         )
         STATISTICS; REPET; VAL; <min>;<prob>;<max>;<moyenne>;↵
         (STATISTICS; REPET; RAN; <min>;<prob>;<max>;;↵)
         STATISTICS; REPET; DIS; <typ-dist>;<paramètre>;↵
    )
```

```
( GROUP; <niveau>;<num-ord>;<nom-gr>;<date-d-maj>;┘
    (IMAGE;<type>;<nom>;┘)
    (ATTRIBUTE;<nom-at>;┘
        (STATISTICS;AT;VAL;<chaîne 30>;┘)
        (STATISTICS;AT;RAN;<min>;<prob>;<max>;┘)
        STATISTICS;AT;DIS;<typ-dist>;<paramètre>;┘
    )
    STATISTICS; REPET; VAL; <min>;<prob>;<max>;<moyenne>;┘
    (STATISTICS; REPET; RAN; <min>;<prob>;<max>;;┘)
    STATISTICS; REPET; DIS; <typ-dist>;<paramètre>;┘
    ( ELEMENT; <niveau>;<num-ord>;<nom-el>;<date-d-maj>;<typ-form>;
              <nbre-av-virgule>;<nbre-ap-virgule>;┘
        (IMAGE;<type>;<nom>;┘)
        (ATTRIBUTE;<nom-at>;┘
            (STATISTICS;AT;VAL;<chaîne 30>;┘)
            (STATISTICS;AT;RAN;<min>;<prob>;<max>;┘)
            STATISTICS;AT;DIS;<typ-dist>;<paramètre>;┘
        )
        STATISTICS; REPET; VAL;<min>;<prob>;<max>;<moyenne>;┘
        (STATISTICS; REPET; RAN; <min>;<prob>;<max>;;┘)
        STATISTICS; REPET; DIS; <typ-dist>;<paramètre>;┘
    )
)
```

```
    ( ROLE; <nom-rol>;<nom-ent>;<date-d-maj>;↵
        (IMAGE;<type>;<nom>;↵)
        (ATTRIBUTE;<nom-at>;↵
              (STATISTICS;AT;VAL;<chaîne 30>;↵)
              (STATISTICS;AT;RAN;<min>;<prob>;<max>;↵)
              STATISTICS;AT;DIS;<typ-dist>;<paramètre>;↵
         )
        STATISTICS; CARD; VAL; <min>;<prob>;<max>;<moyenne>;↵
        (STATISTICS; CARD; RAN; <min>;<prob>;<max>;;↵)
        STATISTICS; CARD; DIS; <typ-dist>;<paramètre>;↵

       STATISTICS; CON; VAL; <min>;<prob>;<max>;<moyenne>;↵
        STATISTICS; CON; DIS; <typ-dist>;<paramètre>;↵
    )
)

(IDENTIFIER; ENT; <nom-ent>;↵
 IDENTIFIER; REL; <nom-rel>;↵
    ( COMPONENT; ELEM; <nom-el>;↵)
    ( COMPONENT; ROL; <nom-rol>;<nom-ent>;↵)
    ( COMPONENT; GR; <nom-gr>;↵)
)

( DEP_FCT;↵
    IN; <nom-rel>;↵
    (DETERMINED; <nom-rol>;↵)
    (DETERMINES; <npm-rol>;↵)
)
```

b) fichier auxiliaire

```
SYSTEM; <nom-demandeur>;<date-extraction>;↵
SCHEMA; <nom-s>;<date-d-maj>;↵
     DESCRIPTION; SCHEMA; <nom-s>;<description>;↵
     ( SYNONYM; <nom-s>;↵)
( ENTITY; <nom-ent>; <date-d-maj>;↵
    (IMAGE;<type>;<nom>;↵)
    (ATTRIBUTE;<nom-at>;↵
          (STATISTICS;AT;VAL;<chaîne 30>;↵)
          (STATISTICS;AT;RAN;<min>;<prob>;<max>;↵)
          STATISTICS;AT;DIS;<typ-dist>;<paramètre>;↵
     )
     DESCRIPTION; ENTITY; <nom-ent>;<description>↵
     ( SYNONYM; <nom-ent>;↵)
     ( ELEMENT; <niveau>;<num-ord>;<nom-el>;<date-d-maj>;↵
         (IMAGE;<type>;<nom>;↵)
         (ATTRIBUTE;<nom-at>;↵
               (STATISTICS;AT;VAL;<chaîne 30>;↵)
               (STATISTICS;AT;RAN;<min>;<prob>;<max>;↵)
               STATISTICS;AT;DIS;<typ-dist>;<paramètre>;↵
          )
         DESCRIPTION; ELEMENT; <nom-el>;<description>;↵
         ( SYNONYM; <nom-el>;↵)
         (VALUE; CODE; <sequence>; <valeur>; <code>;<prob>;↵)
         (VALUE; DOMAIN; <sequence>;<valeur>;<prob>;↵)
     )
```

```
( GROUP; <niveau>;<num-ord>;<nom-gr>;<date-d-maj>;↵
    (IMAGE;<type>;<nom>;↵)
    (ATTRIBUTE;<nom-at>;↵
        (STATISTICS;AT;VAL;<chaîne 30>;↵)
        (STATISTICS;AT;RAN;<min>;<prob>;<max>;↵)
        STATISTICS;AT;DIS;<typ-dist>;<paramètre>;↵
    )
    DESCRIPTION; GROUP; <nom-gr>;<description>;   ↵
    ( SYNONYM; <nom-gr>;↵)
    ( ELEMENT; <niveau>;<num-ord>;<nom-el>;<date-d-maj>;↵
        (IMAGE;<type>;<nom>;↵)
        (ATTRIBUTE;<nom-at>;↵
            (STATISTICS;AT;VAL;<chaîne 30>;↵)
            (STATISTICS;AT;RAN;<min>;<prob>;<max>;↵)
            STATISTICS;AT;DIS;<typ-dist>;<paramètre>;↵
        )
        DESCRIPTION; ELEMENT; <nom-el>;<description>;↵
        ( SYNONYM; <nom-el>;↵)
        (VALUE; CODE; <sequence>; <valeur>; <code>;<prob>;↵)
        (VALUE; DOMAIN; <sequence>;<valeur>;<prob>;↵)
    )
)
```

```
( RELATION; <nom-rel>; <date-d-maj>;↵
    (IMAGE;<type>;<nom>;↵)
    (ATTRIBUTE;<nom-at>;↵
          (STATISTICS;AT;VAL;<chaîne 30>;↵)
          (STATISTICS;AT;RAN;<min>;<prob>;<max>;↵)
          STATISTICS;AT;DIS;<typ-dist>;<paramètre>;↵
    )
    DESCRIPTION; RELATION; <nom-rel>;<description>;↵
    ( SYNONYM; <nom-rel>;↵)
    ( ELEMENT; <niveau>;<num-ord>;<nom-el>;<date-d-maj>;↵
        (IMAGE;<type>;<nom>;↵)
        (ATTRIBUTE;<nom-at>;↵
              (STATISTICS;AT;VAL;<chaîne 30>;↵)
              (STATISTICS;AT;RAN;<min>;<prob>;<max>;↵)
              STATISTICS;AT;DIS;<typ-dist>;<paramètre>;↵
        )
        DESCRIPTION; ELEMENT; <nom-el>;<description>;↵
        ( SYNONYM; <nom-el>;↵)
        (VALUE; CODE; <sequence>; <valeur>; <code>;<prob>;↵)
        (VALUE; DOMAIN; <sequence>;<valeur>;<prob>;↵)
    )
    ( GROUP; <niveau>;<num-ord>;<nom-gr>;<date-d-maj>;↵
        (IMAGE;<type>;<nom>;↵)
        (ATTRIBUTE;<nom-at>;↵
              (STATISTICS;AT;VAL;<chaîne 30>;↵)
              (STATISTICS;AT;RAN;<min>;<prob>;<max>;↵)
              STATISTICS;AT;DIS;<typ-dist>;<paramètre>;↵
        )
        DESCRIPTION; GROUP; <nom-gr>;↵
        ( SYNONYM; <nom-gr>;↵)
```

```
      ( ELEMENT; <niveau>;<num-ord>;<nom-el>;<date-d-maj>;↵
          (IMAGE;<type>;<nom>;↵)
         (ATTRIBUTE;<nom-at>;↵
                (STATISTICS;AT;VAL;<chaîne 30>;↵)
               (STATISTICS;AT;RAN;<min>;<prob>;<max>;↵)
              STATISTICS;AT;DIS;<typ-dist>;<paramètre>;↵
       )
            DESCRIPTION; ELEMENT; <nom-el>;<description>; ↵
            ( SYNONYM; <nom-el>;↵)
            (VALUE; CODE; <sequence>; <valeur>; <code>;<prob>;↵)
            (VALUE; DOMAIN; <sequence>;<valeur>;<prob>;↵)
      )
   )
   ( ROLE; <nom-rol>;<nom-ent>;<date-d-maj>;↵
       (IMAGE;<type>;<nom>;↵)
       (ATTRIBUTE;<nom-at>;↵
            (STATISTICS;AT;VAL;<chaîne 30>;↵)
            (STATISTICS;AT;RAN;<min>;<prob>;<max>;↵)
           STATISTICS;AT;DIS;<typ-dist>;<paramètre>;↵
       )
       DESCRIPTION; ROLE; <nom-rol>;<description>;↵
       ( SYNONYM; <nom-rol>;↵)
   )
)
```

Remarques concernant les informations des 2 fichiers

1) Introduction

Les remarques qui suivent, portent principalement sur le contenu de ces différentes informations mais aussi sur la représentation DSL, ainsi que sur certains choix de saisie et de représentation de ces informations.

2) Le format d'un "element"

Comme valeur de l'attribut format, nous ne permettrons que l'emploi des types de format prévus par PROTO.

3) Le nom des distributions :

Les noms permis en DSL seront repris tels quels dans le fichier.

4) L'identifiant

Le ou les identifiants repris au sein du fichier, comprendront uniquement les éléments suivants :

- le ou les objets "element" directs, c'est-à-dire le ou les objets "element" apparaissant dans la relation "consist of" de l'objet identifié.
- le ou les objets "group" directs.
- un ou plusieurs objets "role".

Ces noms de rôle ne seront pas remplacés par l'identifiant de l'objet "entity" qui joue ce rôle, car cela consisterait à faire un choix prématuré d'implémentation.

5) Les sous-types

Pour tout objet "entity" appartenant à une hiérarchie d'objets "entity", nous nous contenterons de signaler l'objet entity "père" sans lui transmettre les propriétés dont il hérite.

6) La connectivité, la cardinalité et la répétitivité

La connectivité, la cardinalité et la répétitivité s'exprimeront uniquement par le biais d'un "sys-par".

Le nom du "sys-par" pourra être significatif en lui-même ou bien n'être considéré que comme le nom d'un objet. Les deux cas pourront aussi se présenter simultanément (cf. chapitre 3, pt. 3.1.2 B).

Dans le cas où le nom de "sys-par" n'est pas significatif, on trouvera les valeurs de la connectivité ainsi que l'expression d'une probabilité, ou d'une moyenne si cela est souhaité, dans une ou plusieurs clauses "VALUE ...", "RANGE ..." ou "AVERAGE...".

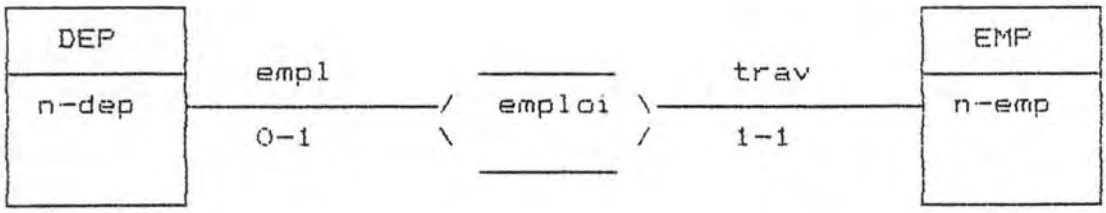
Dans le cas contraire, c'est-à-dire dans le cas ou le nom du "sys-par" est significatif, il correspondra à un format particulier qui permettra d'exprimer une valeur minimale et une valeur maximale. Il est toutefois possible d'exprimer une probabilité ou une moyenne à l'aide d'une relation "RANGE...","AVERAGE..." dans l'objet "sys-par" correspondant au nom donné.

Le format prédéfini du nom du "sys-par" sera le suivant :

sp-<min>-<max>-<identificateur>

L'identificateur doit permettre de distinguer deux objets "sys-par" qui représenteraient une même connectivité, répétitivité ou cardinalité, mais avec des probabilités différentes.

Exemple :



Deux manières de décrire le role "empl" en DSL :

1) Le non du système-paramètre est non significatif .

```
DEFINE  ROLE empl;

ASSUME BY dep IN  emploi WITH CONNECTIVITY c-empl;
     ...

DEFINE SYS-PAR c-empl;

   RANGE VALUE IS 0 WITH PROBABILITY 0.2;
   RANGE VALUE IS 1 THRU N WITH PROBABILITY 0.8;
```

2) Le non du système-paramètre est significatif .

```
DEFINE  ROLE empl;

ASSUME BY dep IN  emploi WITH CONNECTIVITY sp-0-N-emp;
     ...

DEFINE SYS-PAR sp-0-N-empl;

   RANGE VALUE IS 0 WITH PROBABILITY 0.2;
   RANGE VALUE IS 1 THRU N WITH PROBABILITY 0.8;
```

Dans le cas où aucune moyenne n'a été déclarée dans une clause "AVERAGE ...", la présencede plusieurs intervalles nous obligera à calculer la somme des moyennes de chaque intervalle, pondérées par la probabilité associée à chacun d'eux , si aucune probabilité n'est associée aux intervalles, alors la somme des moyennes sera divisée par le nombre d'intervalles.

Exemple :
- - - - -

```
RANGE VALUE IS 1 TO 10 WITH PROBABILITY 0.2;
RANGE VALUE IS 11 TO 14 WITH PROBABILITY 0.4;
RANGE VALUE IS 15  TO 20 WITH PROBABILITY 0.4;

moyenne = moy([1,10])*  0.2 +moy([11,14]) * 0.4
          +moy([15,20]) * 0.4
        = 13.1
```

8) "SAME DOMAIN AS"
- - - - - - - - - - -

Cette relation ne sera pas reprise telle quelle dans l'objet "element" auquel elle correspond mais elle sera remplacée par le domaine correspondant, bien que cela implique, peut-être, dans certains cas, la suppression d'une contrainte de domaine sur l'"element", exprimée par le biais de cette relation.

D.2 Contraintes

1) Contraintes générales.

. Tout nombre doit avoir une longueur inférieure ou égale à 10.

. Tout nombre doit être cadré à gauche.

2) Contraintes portant sur les objets "entity", "relation","role"

A. Contraintes portant sur la cardinalité

1. La cardinalité est obligatoire.

2. Dans le cas où la cardinalité est exprimée au moyen d'un objet "system-parameter" :

   - Si dans une clause "RANGE IS", un intervalle est spécifié avec une probabilité, alors si plusieurs intervalles sont spécifiés une probabilité devra leur être associée et la somme de toutes ces probabilités devra être égale à 1.

   - L'utilisateur ne pourra spécifier qu'une seule clause "AVERAGE VALUE" et/ou "VALUE IS"

   - Si le nom de l'objet "system-parameter" n'est pas significatif (cf. chapitre 3, pt. 3.2.2 B), il conviendra à l'utilisateur de spécifier une valeur maximale au moyen de la clause "VALUE IS", ou de la clause "RANGE IS" en spécifiant un intervalle donnant les valeurs minimale et maximale.

B. Contraintes portant sur la connectivité

1. Présence obligatoire de la connectivité pour tout objet "role".

2. cf. contrainte 2. du point précédent.

3) Contraintes portant sur les objets "element"

A. Un format doit être déclarer pour chaque objet "element", et celui-ci ne peut être qu'un des formats permis par PROTO.

B. Si un code est spécifié au moyen de la relation :

CODE ({sys-par } [{sys-par} THRU {sys-par } : ,) ;
{any-value} {any-value}

où un intervalle [A, B] est spécifié,

on vérifiera que si les valeurs A et B de cet intervalle sont spécifiées au moyen d'objets "system-parameter", aucune clause "RANGE VALUE" n'a été spécifiée mais bien une clause "VALUE IS".

C. Si un domaine de valeurs est spécifié au moyen de la relation :

DOMAIN OF VALUE ARE ({sys-par } [THRU {sys-par }]:,);
{any-value} {any-value}

où un intervalle [A,B] est spécifié,

(cf. contrainte précédente)

D.3 Tests

1) Tests portant sur les objets "entity", "relation" et "role"

tst 1.e : Sélection des objets "entity" pour lesquels il n'existe pas de relation "CARDINALITY IS ..."

Diagnostic :

La spécification des objets sélectionnés est incomplète, aucune cardinalité n'a été spécifiée.

Correction :

La cardinalité des objets sélectionnés doit être spécifiée.

tst 2.e : Sélection des objets "entity" pour lesquels la relation "CARDINALITY IS <sys-par name> ..." <string>

ne possède pas une valeur correcte telle quelle a été définie dans l'annexe D au point 2.A.

Diagnostic :

La spécification de la cardinalité des objets sélectionnés est incorrecte.

Correction :

Les cardinalités des objets sélectionnés doivent être modifiés de manière á obéir aux règles énoncées au point 2.A de l'annexe D.

tst 3.e : idem tst 2.e mais cette fois ci pour la connectivité associée á un objet "role".

tst 4.e : Sélection des objets "relation" pour lesquels aucune relation "IDENTIFY BY ..." n'est spécifiée.

Diagnostic :

Les objets sélectionnés ne possède pas d'identifiant.

Correction :

Pour chaque objet sélectionné, il faut spécifier une clause "IDENTIFY BY ..." spécifiant l'identifiant minimal de cet objet.

2) Tests portant sur les objets "element"

tst 5.e : Sélection des objets "element" pour les quels il n'existe pas de relation "FORMAT IS ...".

Diagnostic :

La spécification des objets sélectionnés est incomplète, aucun format n'est spécifié.

Correction :

Pour chaque objet sélectionné, il faut spécifier un format.

tst 6.e : Sélection des objets "element" pour lesquels il existe une relation "CODE <sys-par name> MEANS ..." et pour lesquels cette relation est incorrecte par rapport aux règles définies au point 3.B de l'annexe D.

Diagnostic :

La spécification du code de chaque objet sélectionné est incorrecte.

Correction :

Pour chaque objet séletionné, il faut corriger le code de manière á ce qu'il réponde aux règles énoncées au point 3.B de l'annexe D.

test 7.e : idem 5.e mais cette fois pour la relation " DOMAIN OF VALUE ... "

ANNEXE E : INTERFACE-UTILISATEUR DU SYSTEME EXPERT POUR LA CONCEPTION DE BASES DE DONNEES

PLAN

Seconde partie : CONTRAINTES DE TYPE DYNAMIQUE.

1. Contraintes intra-relation.

1.1 Contraintes de domaine.

1.1.1 Contrainte individuelle de transition
1.1.2 Contrainte ensembliste de transition
1.1.3 Contrainte transitionnelle multi-rubrique

1.2 Contraintes d'existence.

1.2.1 Presence conditionnelle
1.2.2 Existence conditionnelle

2. Contraintes inter-relations.

2.1 Contraintes de domaine.

2.1.1 Contrainte ensembliste de transition
2.1.2 Contrainte transitionnelle multi-rubrique

2.2 Contraintes d'existence.

2.2.1 Présence conditionnelle

Troisième partie : FORMALISME DES DIFFERENTS TYPES DE CONTRAINTES D'INTEGRITE.

1. Hypothéses simplificatrices.
   .....

2. Eléments de base.
   .....

3. Description des contraintes d'intégrité.
   .....

   3.1 Contraintes de type statique.
   (intra et inter-relation)

   3.1.1 Contraintes de domaines
   3.1.2 Contraintes d'existence

   3.2 Contraintes de type dynamique.
   (intra et inter-relation)

   3.2.1 Contraintes de domaine
   3.2.2 Contraintes d'existence

   3.3 Regroupement des différentes syntaxes.

   3.3.1 Contraintes de domaine.
   3.3.2 Contraintes d'existence.

Quatrième partie : CONTROLE DE COHERENCE ET DEDUCTION

1. Possibilités d'incohérence.

1.1 La cohérence interne.

1.2 La cohérence externe.

1.3 Moments opportuns de déclanchement des tests.

1.4 Algorithme intégrant l'analyse syntaxique & la vérification de la cohérence.

(absent de cette annexe)

1.5 Idées d'implémentation.

1.5.1 Liens entre contraintes.
1.5.2 Stockage des informations.
1.5.3 Liaison rubrique-contraintes.
1.5.4 Transformation d'une contrainte conditionnelle.
1.5.5 Echantillon de contraintes.

2. Déduction d'informations.

2.1 Déduction du type d'une rubrique.

2.2 Déduction du domaine de valeurs d'une rubrique.

INTRODUCTION.

Ce travail a été réalisé dans le cadre de notre stage à l'université de Lyon I dans le laboratoire du professeur Kouloumdjam, avec la collaboration de Momsieur Boulanger et de Monsieur Marie-Sainte, sous la direction de Monsieur Kouloumdjam .

Cette étape constitue le point de départ essentiel pour la réalisation d'un interface-user. En effet, le fait de posséder une typologie des contraintes d'intégrité les plus usuelles, qui sont prises en compte par le système expert de conception de BD, va permettre de définir plus facilement le formalisme des différents types de contraintes d'intégrité, et ainsi, permettre un choix plus aisé d'un langage de dialogue.

Ce rapport ne présente pas une classification exhaustive des CI. car comme il est signalé précedemment, seules les CI. les plus usuelles ont été retenues.

NB. : Tous les exemples de contraintes d'intégrité que nous prenons sont tirés d'un exemple que vous trouverez en fin de cette annexe.
Le travail présenté ici se base sur le modèle conceptuel de FLORY [rm.03].

PREMIERE PARTIE

CONTRAINTES DE TYPE STATIQUE

Première partie : CONTRAINTES DE TYPE STATIQUE

Par contraintes de type statique, nous entendons les propriétes qui doivent être vérifiées à tout moment par les éléments de la base de données.

Nous avons classé les CI. en 2 grandes classes : les contraintes intra-relations abordées au point 1, et les contraintes inter-relations abordées au point 2.

1. contraintes intra-relations

Les contraintes intra-relations sont les contraintes qui ne font intervenir que des rubriques d'une même relation. Celles-ci peuvent être classées de la manière suivante:

1.1 Contraintes de domaine.

1.1.1 Contrainte individuelle :

Contrainte déterminant le domaine de valeurs d'une rubrique soit en extension, soit en intention. Elle ne porte donc que sur une seule rubrique d'une relation.

exemple :

- quantité_livrée est de type entier et est supérieur à 0
- taux_tva a pour domaine (18,5; 33,3)

1.1.2 Contrainte ensembliste :

Contrainte déterminant le domaine de valeurs d'une rubrique en fonction d'un ensemble de valeurs prises par cette même rubrique.

exemple :

. Le chiffre d'affaire mensuel est inférieur ou égal à la moyenne des chiffres d'affaires mensuels multipliés par 1,2 pour un même client

1.1.3 Contrainte multi-rubriques :

Contrainte déterminant le domaine de valeurs d'une rubrique en fonction de la valeur d'une ou plusieurs autres rubriques de la même relation.

exemple :

. La quantité livrée ne peut être supérieure à la quantité commandée.

Remarque: il est toutefois possible de trouver des contraintes recouvrant plusieurs de ces classes.

exemple :

. La prime d'un employé est égale à 5% du salaire moyen des employés, ou la prime d'un employé est égale à 5% de la moyenne des salaires des employés.

Cette contrainte peut être classée dans une classe qui serait une intersection des classes 1.1.2 et 1.1.3 :

contrainte déterminant le domaine de valeurs d'une rubrique en fonction d'un ensemble de valeurs prises par une rubrique de la même relation.

1.2 Contraintes d'existence.

1.2.1 Présence inconditionnelle :

Contrainte déterminant le caractère obligatoire ou facultatif d'une rubrique.

exemple :

. Le numéro de client est obligatoire.

1.2.2 Présence conditionnelle :

Contrainte déterminant le caractère obligatoire d'une rubrique en fonction de la satisfaction d'une condition portant sur une rubrique de cette relation.

exemple :

. Le nom de livraison est obligatoire si ce dernier est différent du nom de facturation.

1.2.3 Existence conditionnelle :

Contrainte portant sur la suppresion de rubrique(s) en fonction d'une condition portant sur une ou plusieurs rubriques de la même relation. Cette contrainte peut être considérée comme un cas particulier des contraintes ensemblistes multi-rubriques.

exemple :

. exemple non trouvé.

2. Contraintes inter-relations

2.1 Contraintes de domaine.

2.1.1 Contrainte ensembliste :

Contrainte déterminant le domaine de valeurs d'une rubrique en fonction d'un ensemble de valeurs prises par une ou plusieurs rubriques d'une autre relation.

exemple :

. La somme des montants des règlements est inférieure ou égale à la somme des montants facturés.

2.1.2 Contrainte multi-rubriques :

Contrainte déterminant le domaine de valeurs d'une rubrique en fonction de la valeur d'une ou plusieurs rubriques d'autres relations.

exemple :

. Le taux de remise appliqué à un article est compris dans un domaine de valeurs déterminé par une condition sur article, catégorie-client, quantité-livrée.

2.2 Contraintes d'existence.

2.2.1 Présence conditionnelle :

Contrainte déterminant le caractère obligatoire ou facultatif d'une rubrique en fonction de la satisfaction d'une condition portant sur une rubrique d'une autre relation.

exemple :

. Le lettrage de facture est obligatoire s'il existe un lettrage règlement correspondant, sinon il est interdit.

2.2.2 Existence conditionnelle :

Contrainte portant sur la suppression d'une rubrique en fonction d'une condition portant sur une ou plusieurs rubriques d'autres relations.

exemple :

. Un client ne peut être archivé s'il existe une facture dont dont le montant payé est inférieur au montant net, ou s'il existe pour ce client une facture non lettrée.

SECONDE PARTIE

CONTRAINTES DE TYPE DYNAMIQUE

Seconde partie : CONTRAINTES DE TYPE DYNAMIQUE.

Par contraintes de type dynamique, nous entendons les propriétés qui doivent être vérifiées lors de l'évaluation de la base de données.

Nous avons classé ces contraintes en deux grandes classes : les contraintes faisant référence à l'état précédent l'état actuel de la base de données, et les contraintes faisant référence à un ou plusieurs états antérieurs à l'état actuel de la base de données. Mais comme ces deux classes sont constituées des mêmes sous-classes, nous avons opté pour une présentation unique pour les deux classes et analogue à la première partie, c'est-à-dire qu'au premier point seront abordées les contraintes intra-relation et au second point, les contraintes inter-relations.

1. Contrainte intra-relation.

1.1 Contraintes de domaine.

1.1.1 Contrainte individuelle de transition :

Contrainte déterminant le domaine de valeurs d'une rubrique en fonction de valeurs antérieures de cette rubrique.

exemple :

. Le montant payé d'une facture ne peut que croître.

1.1.2 Contrainte ensembliste de transition :

Contrainte déterminant l'évolution d'une rubrique en fonction de l'évolution d'un ensemble de valeurs antérieures prises par une rubrique de cette relation.

exemple :

. Non présent.

1.1.3 Contrainte transitionnelle multi-rubriques :

Contrainte déterminant l'évolution d'une rubrique en fonction de l'évolution d'autres rubriques de cette même relation.

exemple :

. Le montant de la prime d'un représentant évolue dans les mêmes proportions que le montant des ventes réalisées au cours des 12 derniers mois.

1.2 Contrainte d'existence.

1.2.1 Présence conditionnelle :

Contrainte déterminant le caractère obligatoire d'une rubrique en fonction de la satisfaction d'une condition portant sur l'évaluation d'autres rubriques de la même relation.

exemple :

. Lorsque le montant du chiffre d'affaire d'un client évolue de plus de X %, la rubrique "remise de fidélité" est obligatoire.

2. Contrainte inter-relations.

2.1 Contraintes de domaine.

2.1.1 Contrainte ensembliste de transition :

Contrainte déterminant l'évolution d'une rubrique en fonction de l'évaluation d'ensemble(s) de valeurs antérieures prises par d'autre(s) rubrique(s) d'autre(s) relation(s).

exemple :

. L'augmentation du montant du fixe d'un représentant est fonction de l'évolution de la moyenne des montants des primes annuelles des deux dernières années.

2.1.2 Contrainte transitionnelle multi-rubrique :

Contrainte déterminant l'évolution d'une rubrique en fonction de l'évolution d'une rubrique d'une autre relation.

exemple :

. L'évolution du découvert autorisé d'un client est fonction de l'évolution de son chiffre d'affaire mensuel.

2.2 Contrainte d'existence.

2.2.1 Présence conditionnelle :

Condition déterminant le caractère obligatoire d'une rubrique en fonction de la satisfaction d'une condition portant sur l'évolution d'une rubrique d'une autre relation.

exemple :

. La rubrique "liste rouge" est obligatoire si la somme des montants dus sur 3 mois est supérieure à son découvert et varie positivement.

TROISIEME PARTIE

FORMALISME DES DIFFERENTS TYPES DE CONTRAINTES D'INTEGRITE

Troisième partie : FORMALISME DES DIFFERENTS TYPES DE CONTRAINTES D'INTEGRITE.

Possédant les types de contraintes à prendre en compte, nous allons essayer de définir une syntaxe externe, c'est-à-dire la syntaxe que l'utilisateur devra suivre pour entrer une contrainte.

Cet objectif sera atteint en un certain nombre d'étapes : la première de celles-ci consistera à définir une syntaxe propre à chaque type de contrainte; la seconde consistera à valider les syntaxes obtenues à la première étape; la troisième étape consistera à regrouper le plus possible les différentes syntaxes afin d'obtenir un nombre réduit de syntaxes.

Cependant cette dernière étape ne doit pas se faire au détriment de la clareté, lisibilité et simplicité de la syntaxe.

Comme nous l'avons déjà signalé lors des deux premières parties,notre recencement n'est pas exhaustif. Il est donc évident qu'il est possible que l'utilisateur désire entrer une contrainte ne rentrant pas dans le formalisme décrit, ou de façon trop complexe. A cette effet une échappatoire devra être prévue.

0. Description de la syntaxe choisie :

- Les mots en majuscule sont des mots réservés.

exemple :
. ET
. OU
. EST DE TYPE : ...

- Les mots en minuscule sont des mots dont les valeurs sont choisies par l'utilisateur.

exemple :
. rubrique
. constante
. opérateur ...

- Les mots entourés de [] sont des clauses optionnelles à l'intérieur d'une instruction.

exemple :
. <rubrique> ::= <string>[:<texte-libre>].

- Les mots entourés de () sont des clauses répétitives.

exemple :
. (<dep-fonct>[,]) DETERMINE[NT] (...).

- Les mots entourés de {} et disposés en colonnes ne peuvent être utilisés simultanément, mais au moins un des mots doit être utilisé.

exemple :
. <réel> ::= {+} <entier> [,<entier>].
{-}

- Les caractères entourés de "" doivent être considéré comme des mots réservés faisant partie de la syntaxe.

exemple :

. ... <agrégat>"("<rubrique>")" ...

1. Hypothèses simplificatrices.

Afin de simplifier la conception de notre première maquette, un certain nombre d'hypothèses ont été posées :

- Les rubriques portent un nom unique.

Conséquence : seul le nom d'une rubrique sera retenu pour déterminer celle-ci. Tout autre élément est superflu.

- Pour les contraintes dynamiques, on parlera uniquement en terme d'état(s) antérieur(s).

Conséquence : absence de la notion explicite de temps.

2. Elements de bases.

```
<entier> ::= 0/1/2/3/4/5/6/7/8/9:/<entier><entier>.

<carartère> ::= a/b/c/d/e/f/g/h/i/j/k/l/m/n/o/p/q/r/s/t/u/v/w/x/y
                /z/A/B/C/.../X/Y/Z/%/#/&/1/2/3/4/5/6/7/8/9/0.
<chaine-car> ::= <caractère>/<chaine-car><caractère>.

<réel> ::= {+}<entier>[,<entier>].
           {-}

(Non réalisable pour l'instant en "Foll Prolog".)


<constante>::= <caractère>/<entier>/<réel>
               /<constante><caractère>.


<rubrique> ::= <chaine-car>[=<texte-libre>].

<label> ::= <chaine-car>":".

<comp> ::= </>/<=/>=/=/<>.

<agrégat> ::= min/max/moyenne/somme/cardinalite.

<oper> ::= +/-/*/:.

<type-pred> ::= entier/alphabétique/alpha-numérique
                /réel/booléen.


<dep-fonct> ::= (<rubrique>[,]) DETERMINE[NT] (<rubrique>[,]).


<formule>  ::= [(<oper>{<entier>                    })].
                       {<réel>                      }
                       {<rubrique>                  }
                       {<agrégat> "("<rubrique>")"}

<formulet>::= [(<oper>

        {<entier>                                                  })].
        {<réel>                                                    }
        {<rubrique>["["([-]<entier>[{:}])"]"]                      }
        {                              {,}                         }
        {<agrégat>"("<rubrique>["["([-]<entier>[{:}])"]"]")"}
                                                 {,}
```

3. Description des contraintes d'intégrité.

3.1 Contraintes de type statique.

(intra et inter-relations)

3.1.1 Contraintes de domaine.

- Contrainte individuelle :

```
SDI ::= <label><rubrique> { EST  DE TYPE :<type-pred> [ET]          }
                          { A POUR DOMAINE : (<constante>[;]) [OU]}

       [(<comp><constante>)[ET]].
                           [OU]
```

- Contrainte ensembliste :

```
SDE   ::= <label><rubrique>(<comp>{<réel>                    }
                               {<entier>                     }
                               {<agrégat>"("<rubrique>")"}

      [(<oper> {<réel>                    })] [ET]).
               {<entier>                   }      [OU]
               {<agrégat>"("<rubrique>")"}
```

- Contrainte multi-rubriques :

```
SDM::= <label>[(SI<rubrique><comp>{<constante>                 }<formule>
                                  {<rubrique>                 }
                                  {<agrégat>"("<rubrique>")"}

      [ET]) ALORS] <rubrique>(<comp> {<constante>                  }
      [OU]                           {<rubrique>                   }
                                     {<agrégat>"("<rubrique>")"}

      <formule>[ET]).
               [OU]
```

3.1.2 Contrainte d'existence.

- Présence inconditionnelle :

```
CSEI ::= <label> <rubrique> EST {OBLIGATOIRE}.
                                {FACULTATIVE}
```

- Présence conditionnelle :

```
CSEC ::= <label><rubrique>EST {OBLIGATOIRE} (SI{<rubrique>                  }
                              {INTERDIT    } (  {<constante>                }
                                                {<agrégat>"("<rubrique>")"}

      <comp> {<rubrique>                  }<formule>[ET]).
             {<constante>                 }         [OU]
             {<agrégat>"("<rubrique>")"}
```

3.2 Contraintes de type dynamique.

(intra et inter-relations)

3.2.1 Contraintes de domaine.

- Contrainte individuelle de transition :

```
CDDI ::= <label><rubrique>["["0"]"](<comp><rubrique>"["[-]<entier>"]"

       [(<oper>{<réel>                       })][ET]).
               {<entier>                     }  [OU]
               {<rubrique>"["[-]<entier>"]"}
```

- Contrainte ensembliste de transition :

```
CDDE ::= <label><rubrique>["["0"]"](<comp>
             <agrégat>"("<rubrique>"["([-]<entier>[{:}])"]"
                                                  {,}

      <formulet>[ET]).
                [OU]
```

- Contrainte multi-rubriques de transition :

```
CDM ::= <label>[(SI <rubrique>["["([-]<entier>[{:}])"]"]<comp>
                                                    {,}

        {<constante>                                                    }
        {<rubrique>["["([-]<entier>[{:}])"]"]                           }
        {                              {,}                              }
        {<agrégat>"(" <rubrique>["["([-]<entier>[{:}])"]"]")"}
                                                    {,}

        <formulet>[ET]) ALORS] <rubrique> ["["0"]"] (<comp>
                  [OU]

        {<constante>                                                    }
        {<rubrique>["["([-]<entier>[{:}])"]"]                           }
        {                              {,}                              }
        {<agrégat>"(" <rubrique>["["([-]<entier>[{:}])"]"]")"}
                                                    {,}

        <formulet>[ET]).
                  [OU]
```

3.2.2 Contrainte d'existence.

- Présence conditionnelle de transition :

```
CDEC ::= <label><rubrique>["["0"]"] EST OBLIGATOIRE ( SI

        {<constante>                                                    }<comp>
        {<rubrique>["["([-]<entier>[{:}])"]"]                           }
        {                              {,}                              }
        {<agrégat>"(" <rubrique>["["([-]<entier>[{:}])"]"]")"}
                                                    {,}

        {<constante>                                                    }<formulet>
        {<rubrique>["["([-]<entier>[{:}])"]"]                           }
        {                              {,}                              }
        {<agrégat>"(" <rubrique>["["([-]<entier>[{:}])"]"]")"}
                                                    {,}

        [ET]).
        [OU]
```

3.3 Regroupement des différentes syntaxes.

3.3.1 Contraintes de domaine.

La contrainte de domaine individuelle du point 3.1.1 sera conservée.

Les autres pourront se ramener sous la forme de la contrainte multi-rubrique de transition du point 3.2.1, si l'on admet qu'une rubrique non suivie d'un intervalle ou d'une suite d'état est à l'état 0.

3.3.2 Contrainte d'existence.

Pour les contraintes d'existence, la contrainte de présence conditionnelle de transition du point 3.2.2 est la plus générale, si à la place de OBLIGATOIRE, on a :

{ OBLIGATOIRE }
{ FACULTATIF }
{ INTERDIT }.

QUATRIEME PARTIE

CONTROLE DE COHERENCE
&
DEDUCTION

Quatrième partie : CONTROLE DE COHERENCE & DEDUCTION

Au cours de ce chapitre, dans un premier temps, nous allons essayer de détecter les possibilités d'incohérence pouvant apparaître, de la façon la plus exhaustive possible. La seconde étape consistera à se demander le moment opportun de déclanchement des tests de vérification de la cohérence afin d'éviter des analyses inutiles d'un certain nombre de contraintes.

Nous proposerons ensuite un algorithme intégrant l'analyse syntaxique et la vérification de la cohérence des contraintes pour terminer par quelques réflexions sur la manière possible d'implémentation de certains éléments spécifiés dans ce chapitre.

1. Possibilités d'incohérence.

Nous allons considérés les tests a effectuer en fonction des possibilités d'incohérence pour détecter celles-ci.

Nous classerons ces tests en 2 catégories :

La cohérence interne :

Nous parlons de cohérence interne lorsque les informations nécessaires à la vérification de la cohérence se trouvent regroupés au sein de la contrainte en cours d'analyse ou d'informations autres que les contraintes déjà présentes dans le système.

La cohérence externe :

Nous parlons de cohérence externe lorsque les informations nécessaires à la vérification de la cohérence se trouvent dans une ou plusieurs contraintes déjà présentes dans le système.

1.1 Tests de cohérence interne.

(tst1) Une contrainte ne peut porter que sur des rubriques déjà déclarées.

(tst2) Seule une rubrique peut précéder une liste ou un intervalle d'états.

(tst3) Unicité des noms de rubrique.

(tst4) Dans une contrainte, la rubrique contrainte ne peut apparaître à l'état zéro dans la condition.

(tst5) Dans une contrainte, à l'intérieur d'une conjonction, il est interdit d'égaler plusieurs valeurs à une même rubrique dans le même état.

(tst6) La définition d'un ensemble de valeurs pour une rubrique au moyen des comparateurs <,>,=, liés par des opérateurs logiques ET, ne peut fournir un ensemble vide.

(tst7) Dans une comparaison, il est interdit de trouver de part et d'autre du comparateur la même rubrique dans le même état.

(tst8) Une rubrique contrainte ne peut être qu'à l'état zéro.

(tst9) Toute comparaison ne peut être toujours évaluée à vrai ou à faux.

1.2 Tests de cohérence externe.

(tst10) A l'exception de l'agrégat "cardinalité", tout autre agrégat ne peut porter que sur une rubrique de type entier (ou réel).

(tst11) Toute comparaison ne peut se faire qu'entre éléments de même type.

(tst12) Toute opération arithmétique ne peut porter que sur des opérandes de type entier (ou réel).

(tst13) Unicité des noms de label.

(tst14) Pour une même rubrique, il est interdit d'avoir à la fois :

(1) une présence obligatoire sans condition et facultative
(2) une présence obligatoire sans condition et obligatoire avec condition
(3) une présence obligatoire sans condition et interdite selon certaines conditions
(4) une présence obligatoire avec condition et interdite pour la même condition
(5) une présence facultative et interdite selon certaines conditions

(tst15) Il ne peut y avoir plusieurs contraintes définissant le type et le domaine de valeurs pour une même rubrique.

(tst16) Si pour une rubrique, son domaine de valeurs a été spécifié, alors tout ensemble de valeurs attribué à cette rubrique, dans toute autre contrainte doit avoir une intersection commune avec cet ensemble de valeurs.

(tst17) Deux contraintes ayant une condition identique, ne peuvent avoir une résultante assignant à une même rubrique des ensembles de valeurs différents.

NB. :

Dans le cas où les conditions de plusieurs contraintes définiraient des ensembles de valeurs ayant une intersection non vide, contraintes qui assignent des ensembles de valeurs disjoints à une même rubrique, il convient de signaler ce fait car il constitue une possibilité d'incohérence lors de l'exploitation de ces contraintes.

(tst18) Pour une même condition, il est interdit d'assigner une valeur à une rubrique et en même temps interdire sa présence.

NB. :

A nouveau, il se peut que deux contraintes aient une condition définissant des domaines de valeurs ayant une intersection non vide, tout en ayant pas une condition identique mais se trouvant dans le cas énoncé précédemment. Il convient également de signaler cette possibilité incohérence.

(tst19) Toute contrainte de domaine identique à une autre sera considérée comme redondante et donc supprimée.

(tst20) Toute valeur assignée à une rubrique doit être du type de cette dernière.

1.3 Moments opportuns de déclanchement des tests.

Nous allons maintenant préciser le moment optimal de déclanchement de ces différents tests, permettant ainsi de réduire les coûts de traitement liés à une contrainte.

TESTS	MOMENT(S) DE DECLANCHEMENT
tst1	La syntaxe définissant les lieux possibles de présence d'une rubrique, nous déclancherons donc ce test lors de l'analyse syntaxique.
tst2	idem
tst3	Lors de la déclaration d'une rubrique.
tst4	Lors de l'analyse syntaxique, après obtention de tous les éléments nécessaires à ce test.
tst5	Lors de l'analyse d'une conjonction.
tst6	Après l'analyse d'une conjonction ET SI le domaine de valeurs des rubriques est connu.
tst7	Lors de l'analyse du terme comparé et après l'obtention de tous les éléments nécessaires à ce test.
tst8	Lors de l'analyse de la rubrique contrainte.
tst9	Après chaque comparaison ET SI le domaine de valeurs des rubriques est connu.
tst10	Lors de l'analyse syntaxique d'un agrégat.
tst11	Lors de l'analyse syntaxique de la comparaison.

TESTS	MOMENT(S) DE DECLANCHEMENT
tst12	Lors de l'analyse d'une opération.
tst13	Lors de l'analyse des différents types donnés, en cas d'analyse d'une contrainte.
tst14	Lorsque la rubrique contrainte est connue ainsi que le type exact de présence, pour (1),(2),(3),(5) alors que (4) sera effectué après l'analyse d'une conjonction.
tst15	Après l'analyse d'une conjonction ET de la résultante de la contrainte.
tst16	Après l'analyse d'une conjonction dans une contrainte autre que de domaine.
tst17	Après l'analyse d'une conjonction ET de la résultante de la contrainte.
tst18	idem
tst19	Lors de l'analyse d'une contrainte de domaine pour une conjonction.
tst20	Après analyse d'une assignation.

Remarque :

La plupart des tests peuvent donc être effectués lors de l'analyse syntaxique de la contrainte, permettant ainsi de limiter le nombre de manipulations de cette contrainte.

C'est pourquoi nous nous proposons d'intégrer ces tests à l'analyse syntaxique.

1.5 Idées d'implémentation.

1.5.1 Liens entre contraintes :

Une contrainte sera transformée en plusieurs clauses si celle-ci possède au moins une connection logique OU. Ces clauses ayant même tête de clause et même label seront reliées implicitement par un OU.

Par contre, toutes les contraintes ayant une tête de clause différente sont reliées entre elles par un ET implicite.

1.5.2 Stockage des informations :

Pour conserver les informations concernant une vue, il sera nécessaire de créér au moins deux banques : la première permettant de conserver les informations dans la syntaxe externe pour un dialogue aisé avec l'utilisateur et la seconde permettant le stockage des informations transformée dans la syntaxe interne.

1.5.3 Liaison rubrique-contraintes :

Que ce soit pour la vérification de la cohérence ou pour l'exploitation des contraintes transformées, il est à notre avis, nécessaire de pouvoir retrouver rapidement toutes les contraintes portant sur une certaine rubrique, que celle-ci se trouve reprise dans la condition ou quelle soit la rubrique contrainte.

A cette fin, nous proposons une solution demandant la création d'une nouvelle banque de clauses où sera intégré pour chaque rubrique une clause ayant comme paramètres d'une part le nom de la rubrique en question et une liste contenant le label des contraintes portant sur cette rubrique, le label servant de clé pour accéder à une contrainte.

Cette solution n'est certainement pas optimale, mais elle a l'avantage de permettre une implémentation rapide de l'algorithme mais l'utilisation d'un dictionnaire de données n'est pas à négliger.

1.5.4 Transformation d'une contrainte conditionnelle : .....

Lorsque nous sommes en présence d'une contrainte comportant une condition, il conviendra de la transformer de telle manière que la clause provenant de cette transformation ne soit pas évaluée à faux dans le cas où la condition n'est pas vérifiée.

Lorsque la contrainte sera transformée en plusieurs clauses, seules la dernière clause devra être transformée de cette manière, sinon seule la première des clauses ayant même tête et même label sera exploitée car évaluée à vrai.

```
ex : c1 : SI a alors b
     OU
     c1 : SI c alors d
                 sinon réussi
```

1.5.5 Echantillon de contraintes :

Afin de limiter la durée de la vérification de la cohérence, il serait utile de pouvoir déterminer un échantillon, c'est-à-dire le plus petit sous-ensemble possible de contraintes sur lequel sera réellement effectuée la vérification de cohérence et qui soit tel que lorsque le test de cohérence est vérifié sur cet échantillon, nous puissions affirmer qu'il est également vérifié sur l'ensemble des contraintes d'intégrité.

Dans ce but, nous allons voir s'il est possible de détermier pour chaque test, un échantillon de ce type :

TESTS	ECHANTILLON POSSIBLE
tst1	Seule les clauses de déclaration des rubriques intervenant dans la contrainte analysée sont utiles.
tst2	idem
tst3	idem
tst4	----
tst5	idem
tst6	idem
tst7	idem
tst8	idem
tst9	Seules les clauses de domaine des rubriques intervenant dans la comparaison. Si au moins une de ces clauses est absente, ce test sera reporté à l'introduction de la clause absente et dès lors l'ensemble des clauses portant sur la rubrique dont le domaine vient d'être défini sera nécessaire.
tst10	Seule la clause exprimant le type de la rubrique concernée est utile.
tst11	Clauses définissant le type des rubriques concernées.
tst12	idem
tst13	Clause éventuelle ayant comme paramètre ce même label.

TESTS	ECHANTILLON POSSIBLE
tst14	Clauses d'existence ayant la même rubrique contrainte que la contrainte en cours d'analyse. De plus, si la clause en construction est de type "INTERDIT SI", les contraintes de domaine ayant la même rubrique contrainte sont nécessaires.
tst15	Contraintes de domaine individuelles "EST DE TYPE" et "A POUR DOMAINE" portant sur la même rubrique.
tst16	Contraintes définissant le domaine de valeurs de la rubrique concernée.
tst17	Contraintes conditionnelles définissant assignant un ensemble de valeurs à la même rubrique.
tst18	Clauses d'existence ayant la même rubrique contrainte que la contrainte en cours d'analyse. De plus, si la clause en construction est de type "INTERDIT SI", les contraintes de domaine ayant la même rubrique contrainte sont nécessaires.
tst19	Clauses de même type portant sur cette même rubrique.
tst20	Clause définissant le type de la rubrique concernée.

Remarque :

A l'échantillon défini pour chaque test, il convient naturellement d'ajouter la clause en cours de formation.

A partir de ce tableau, il est assez facile de déduire les tests pouvant demander un traitement assez long, c'est-à-dire les tests 9,14 et 18.

D'autre part, afin de faciliter l'échantillonnage, il serait intéressant de donner à chaque clause créée à partir d'une contrainte, une tête de clause reflètant exactement le type de la contrainte. De cette manière, il sera possible de faire une sélection supplémentaire parmi l'ensemble des clauses portant sur une rubrique (cf. 1.5.3).

2. Déduction d'informations.

Si l'on regarde bien attentivement les règles de cohérence introduites au point 1 de cette quatrième partie, on peut remarquer qu'à partir de certaines d'entre elles, complètées par des informations présentes dans le système, il est possible de déduire d'autres informations.

Nous allons tenter d'expliquer toutes ces possibilités, une manière possible de les implémenter, les problèmes pouvant surgir de ces déductions et si possible les solutions à ces problèmes.

2.1 Déduction du type d'une rubrique.

Parmi l'ensemble des règles, nous avons déclaré que toute comparaison doit être faite entre éléments de même type (cf. tst11) .
De ce fait, il est possible de déduire le type, non spécifié, d'une rubrique, et cela à partir du type de l'autre élément comparé.

Ce principe de déduction est donc assez simple, mais cela devient beaucoup moins évident lorsque les deux éléments comparés n'ont pas encore de type déclaré !!.

Dans ce cas, un type inconnu sera créé pour chacune de ces rubriques ainsi qu'une clause supplémentaire indiquant que ces deux rubriques sont de même type.

Lors de la vérification du test 11, si un type inconnu est trouvé pour une rubrique, tandis que nous disposons du type de l'autre élément comparé, il est possible alors de déduire le type de cette rubrique. Il suffira de remplacer le type inconnu par le type correspondant. Ensuite, il conviendra de répercuter cette déduction vers tout autre rubrique étant déclarée comme ayant le même type que la première citée. A cet instant, la clause spécifiant l'égalité de type entre rubriques devient inutile et peut donc être supprimée.

Passons maintenant à un autre problème. Losrque l'utilisateur rentre le type d'une rubrique pour laquelle ce dernier a été déduit ou est inconnu.

Dans le cas où le type de la rubrique était inconnu, cela revient au cas de figure précédent, mais lorsqu'il s'agit d'une rubrique dont le type a été déduit, deux cas peuvent se présenter :

1) Le type déduit correspond au type spécifié:

Dans ce cas,la seule démarche possible consiste à signaler au système que le type n'est plus déduit mais a été spécifié par l'utilisateur.

2) Le type spécifié diffère du type spécifié :

Il convient de signaler à l'utilisateur qu'il existe une incohérence, ainsi que le chemin de déduction du type de cette rubrique.

Jusqu'à présent, nous avons parlé de type déduit ou spécifié, sans toutefois définir la manière de reconnaître une information déduite d'une information spécifiée. Ce sera donc le point que nous nous proposons d'aborder sans plus attendre.

La solution la plus élégante que nous avons trouvé, consiste à mettre comme paramètres dans la tête de la clause, représentant l'information déduite, le label de la clause ayant permis la déduction ainsi qu'un niveau de déduction.

Lorsqu'il s'agit d'une information déduite, ce niveau de déduction sera supérieur à zéro tandis que pour toute information spécifiée, il sera nul.

Le fait de passer comme paramètre le label de la clause ayant permis une déduction permet de fournir tous les renseignements sur le chemin de déduction. En effet, il est désormais possible de remonter vers la clause ayant permis une déduction car comme nous l'avons déjà signalé, la clé d'accès à une clause est constitué par son label.

2.2 Déduction du domaine de valeurs d'une rubrique.

Parmi l'ensemble des règles de cohérence énoncées précédemment, un certain nombre d'entre elles, complètées par les contraintes déjà existantes, permettent de déduire le domaine de valeurs d'une rubrique. Il s'agit des tests 6,9,16.

Le principe de déduction n'est pas aussi aisé que celui évoqué au point précédent. Pour vous en persuader, nous allons vous présenter les différents cas de figure pouvant se présenter :

A) Si l'on est pas dans la cas d'une contrainte de domaine :

1) Si la comparaison analysée se fait entre deux rubriques :

1.1)

Si le domaine des deux rubriques n'est ni spécifié, ni déduit, aucune déduction n'est possible, seul le lien existant entre ces deux rubriques peut être mémorisé. Lorsque le domaine d'une des deux rubriques sera déduit ou spécifié,la déduction du second sera rendue possible par cette mémorisation antérieure.

1.2)

Si le domaine de valeurs d'une seule des rubriques comparées est connu, il convient de déduire le second en fonction du comparateur, c'est-à-dire de telle manière que la comparaison puisse être évaluée à vrai, tout en sachant qu'il ne s'agit que de la déduction d'un domaine possible et partiel.

2) Si la comparaison en cours d'analyse se fait entre une rubrique et une constante :

Dans ce cas, la déduction d'un domaine possible et partiel est évisente.

B) Dans le cas d'une contrainte de domaine :

On réalise les mêmes opérations que dans le point A si la variable comparée possède un domaine non déduit ou si l'on compare à une constante. Mais les déductions réalisées formeront non plus un domaine déduit mais le domaine proprement dit de la variable.

Ayant déduit un domaine de valeurs pour une rubrique, voyons à quels problèmes nous pouvons être confrontés :

1) Si l'utilisateur spécifie un domaine de valeurs pour cette rubrique,il convient de s'assurer qu'il n'y aie pas d'incohérence avec le domaine déduit.

   Comme le domaine déduit n'est qu'un domaine partiel, nous dirons qu'il n'y a pas d'incohérence s'il existe intersection non vide entre ce domaines.

   En effet,si ces domaines sont disjoints, nous pouvons affirmer que le domaine spécifié apportera de l'incohérence vu que le test 9 ne sera pas vérifié pour la comparaison ayant permis la déduction du domaine de valeurs.

   Dans le cas où il existe une intersection non vide entre ces domaines,il convient de remplacer le domaine déduit par le domaine spécifié. De plus,si le domaine spécifié n'est pas inclus totalement dans le domaine déduit,nous sommes assurés de l'absence d'incohérence, vu que le seul test restant, c'est-à-dire le 9 sera toujours vérifié.

   Par contre,dans le cas où le domaine déduit contient le domaine spécifié,nous sommes assurés que le test 9 ne sera pas vérifié pour la comparaison ayant permis la déduction et donc que le domaine spécifié amène de l'incohérence.

   D'autre part, s'il y a modification, celle-ci doit être répercutée vers les domaines de valeurs déduits à partir de ce dernier.

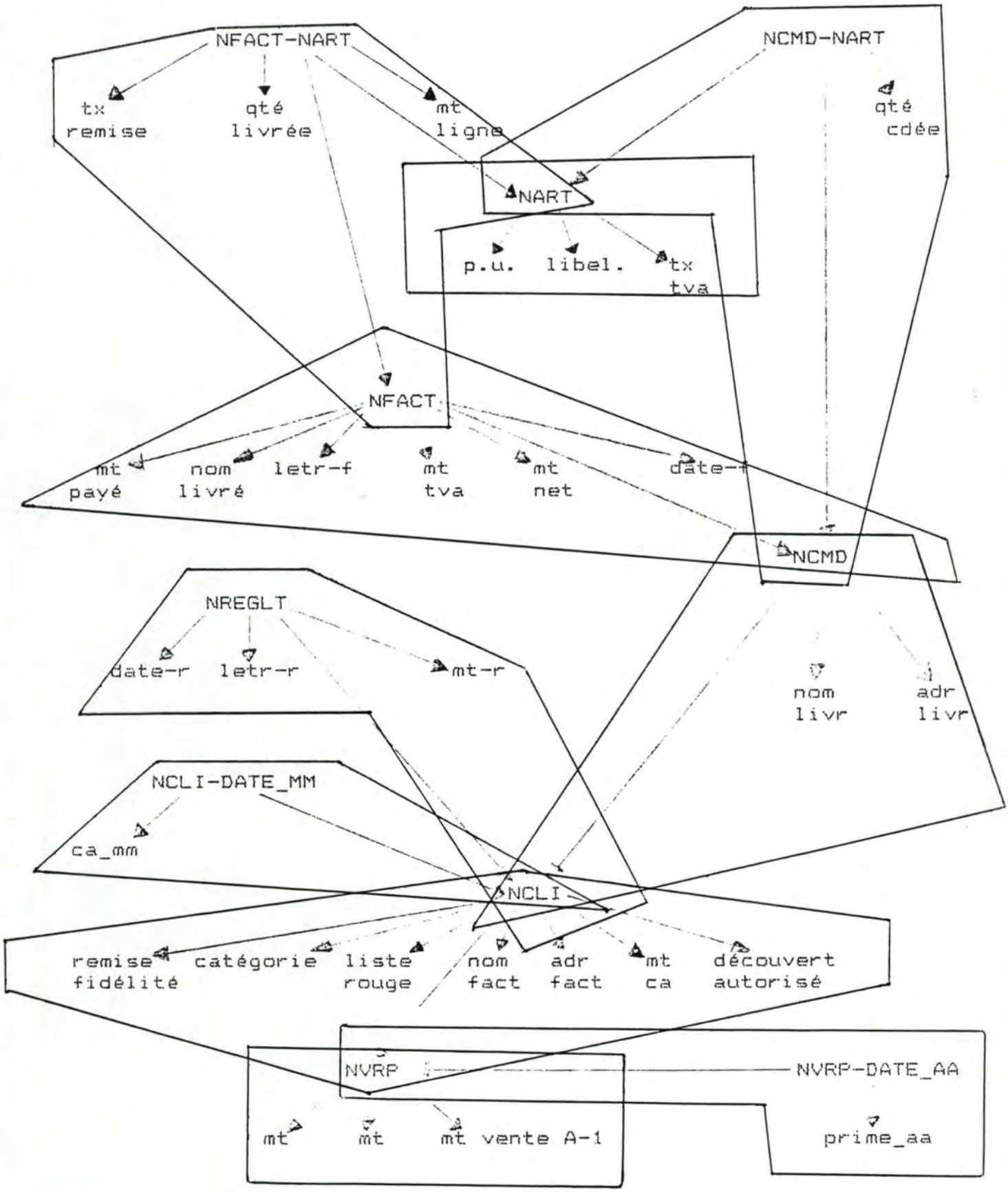
2) Si lors de l'analyse d'une comparaison,il s'avère qu'un des éléments comparés est une rubrique dont le domaine de valeurs a été déduit, domaine n'étant que partiel, il convient d'effectuer une nouvelle déduction en fonction de cette comparaison.
   Si cette nouvelle déduction est identique à la première, elle peut être abandonnée. Dans tous les autres cas, elle devra être conservée et sera liée à la première par un ET logique car la perte de toute information ne nous permettra plus de faire des conclusions concernant la vérification du test 9.

Remarque :

Vu la difficulté que représente la déduction d'informations, même si elle est facilitée par les règles de cohérence, vous pourriez vous demander quel était notre but en abordant ce point.

Le but recherché lorsque nous désirons intégrer la déduction d'informations, c'est d'une part de permettre d'éviter de retarder la vérification de la cohérence jusqu'au moment où toutes les informations sont disponibles, car cette vérification sera d'autant plus longue que le nombre de contraintes entrées avant de pouvoir déclancher le test, est grand, et d'autre part, de permettre la vérification plus rapide de certains tests de cohérence qui sans cette déduction allongeraient fortement le temps de réponse du système.

Exemple traité dans ce rapport.



ANNEXE F : Pré-requis pour le bon fonctionnement des différents tests

Comme le titre de cette annexe le suggère, nous allons reprendre les tests du chapitre 5 afin de déterminer pour chacun d'entre eux quelles sont les informations qui doivent être présentes dans la spécification d'un schéma e-a pour qu'ils soient suceptibles de déterminer les écarts pouvant exister pour ce schéma par rapport à la forme canonique définie au chapitre 2 de ce travail.

C'est ce qu'illustrent les tableaux suivants :

F.1 Tests de complétude

TEST N°	PRE-REQUIS
tst 1-c tst 2-c tst 3-c	Il est nécessaire de posséder l'ensemble de la spécification de l'objet "entity"
tst 4-c tst 5-c tst 6-c tst 7-c	Il est nécessaire de posséder l'ensemble de la spécification de l'objet "element"
tst 8-c tst 9-c tst 10-c tst 11-c	Il est nécessaire de posséder l'ensemble de la spécification de l'objet "group"
tst 12-c tst 13-c	Il est nécessaire de posséder l'ensemble de la spécification de l'objet "relation"
tst 14-c	Il est nécessaire de posséder l'ensemble de la spécification de l'objet "role"
tst 15-c tst 16-c	Il est nécessaire de posséder l'ensemble de la spécification de l'objet "aggregate"

F.2 Tests de redondance

TEST N°	PRE-REQUIS
tst 1-r	La relation "IDENTIFIED BY", la relation "FUNCTIONALLY DETERMINES" au sein des objets concernés ("aggregate","element","group")
tst 2-r	La relation "IDENTIFIED bY", la relation "CONSIST OF" de l'objet "aggregate" qui joue le role d'identifiant
tst 3-r	La spécification complète de l'objet "element" considéré
tst 4-r	La spécification complète de l'objet "group" considéré
tst 5-r	La spécification complète de l'objet "relation" considéré
tst 6-r	La relation "IDENTIFIED BY", la relation "FUNCTIONALLY DETERMINES" au sein des objets concernés ("role","aggregate")
tst 7-r	La contrainte d'intégrité concernée
tst 8-c	La relation "FUNCTIONALLY DETERMINES" [plus la connectivité des objets "role" impliqués dans la dépendance fonctionnelle]
tst 9-r	La relation "CONSIST OF" de l'objet "aggregate" concerné

F.3 Tests de cohérence

TEST N°	PRE-REQUIS
tst 1-h	
a)	La relation "IDENTIFIED BY", la relation "CONSIST OF" et la spécification de l'objet "element" ou "group"
b)	La relation "RELATED BY" et la spécification de l'objet "relation" intervenant dans cette relation
c)	a) + b)
tst 2-h	La relation "SUBTYPE OF" et "SUBTYPE ARE" de l'objet "entity" considéré, les relations "SUBTYPE OF" de l'objet "entity" super-type et "SUBTYPE ARE" de(s) objet(s) "entity" sous-type et ainsi de suite.
tst 3-h	La relation "CONSIST OF" de l'objet "entity" considéré
tst 4-h	La relation "FUNCTIONALLY DEPENDENT ON", la relation "CONTAINED IN", la spécification de l'objet "entity" contenant l'objet "element" concerné et les relations "IDENTIFIED BY"
tst 5-h	La relation "CONTAINED ... IN"
tst 6-h	La relation "CONTAINED ... IN" de l'objet "element" considéré et la spécification de l'objet contenant l'objet "element"
tst 7-h	La relation "IDENTIFIED BY" de l'objet "relation" concernée, la relation "RELATES...", éventuellement la spécification de l'objet "aggregate" apparaissant dans "IDENTIFIED BY" de l'objet "relation" et la relation "CONSIST OF" de cet objet "aggregate"
tst 8-h	La relation "IDENTIFIED BY" de l'objet "relation" où apparait un objet "aggregate", la relation "CONSIST OF" de ce dernier et la spécification des objets apparaissant dans cette dernière relation

TEST N°	PRE-REQUIS
tst 9-h	La relation "CONTAINED IN" de l'objet "relation", la relation "RELATES ... WITH CONNECTIVITY" et la spécification des objets "entity" jouant les différents roles avec la relation "CARDINALITY IS ..."
tst 10-h	La relation "RELATES ... WITH CONNECTIVITY", Les dépendances fonctionnelles éventuelles entre objets "role" apparaissant dans les relations "RELATES ..."
tst 11-h	La relation "IDENTIFY", la relation "ASSUMED BY ... WITH CONNECTIVITY" et la relation "RELATES" de l'objet "relation" identifié par l'objet "role" concerné
tst 12-h	La relation "FUNCTIONALLY DEPENDENT ON ..." de l'objet "role" considéré, la relation "ASSUMED BY ... IN..." et la spécification de l'objet déterminant si c'est un objet "role" alors la relation "ASSUMED ... IN" si c'est un objet "aggregate" alors la relation "CONSIST OF" et la relation "ASSUMED ... IN" des objets "role" qui apparaissent dans la relation "CONSIST OF"
tst 13-h	La relation "FUNCTIONALLY DEPENDENT ON ..." de l'objet "role" considéré, la relation "CONSIST OF" et la spécification de l'objet l'objet déterminant si c'est un objet "role" alors la relation "ASSUMED ... IN" si c'est un objet "aggregate" alors la relation "CONSIST OF" et la relation "ASSUMED ... IN" des objets "role" qui apparaissent dans la relation "CONSIST OF"
tst 14-h	La relation "FUNCTIONALLY DEPENDENT ON" et si le déterminant est un objet "element" ou "group" alors la relation "CONTAINED IN" de cet objet Si le déterminant est un objet "aggregate" alors la relation "CONSIST OF", la spécification des constituants et la relation "CONTAINED IN"